NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

DESIGN OF A GRAPHICAL USER INTERFACE
FOR A MULTIMEDIA DBMS:
QUERY MANAGEMENT FACILITY

by

Charles B. Peabody

September, 1991

Thesis Advisor:	Vincent Y. Lum
Thesis Co-Advisor:	C. Thomas Y. Wu

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS37	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION N

11. TITLE (Include Security Classification)
DESIGN OF A GRAPHICAL USER INTERFACE FOR A MULTIMEDIA DBMS: QUERY MANAGEMENT FACILITY (

12. PERSONAL AUTHOR(S)
PEABODY, Charles Brown

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 9/89 TO 9/91	September, 1991	139

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	GUI, MDBMS, Graphical Query Facility, Retrieval, Multimedia Databas Management System, User Interface.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis presents criteria and necessary features by which to evaluate and design a good graphical user interfa (GUI) for a Multimedia Database Management System (MDBMS). This material is also applicable to a tradition DBMS. Included in the thesis is the specification for a Query Management Facility (QMF) for a MDBMS user inte face. The nature and benefits of the GUI environment, requires that we consider GUI concepts early in the user inte face conceptualization and design. In today's DBMS user interfaces, these GUI concepts are for the most part appli as an after-thought. This is a critical mistake. Early incorporation of GUI capabilities along with established user i terface principles results in a superior user interface. The QMF presented herein is one such interface. It combines t ideas of simple operations and data flow to allow the user to specify his query. Additional concepts used include: pi ture of the database schema, picture of the developing query, selectable objects, direct manipulation, piecemeal que specification, display of intermediate results and pre-defined joins. The resulting QMF is simple to use and enabl the flexible expression of the simple as well as the complex database query.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Vincent Y. Lum	22b. TELEPHONE (Include Area Code): (408) 646-2175	22c. OFFICE SYMBOL: CSLm

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED

Approved for public release; distribution is unlimited.

DESIGN OF A GRAPHICAL USER INTERFACE FOR A MULTIMEDIA DBMS : QUERY MANAGEMENT FACILITY

by

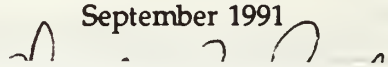
Charles B. Peabody
Captain, United States Marine Corps
B.S., University of New Hampshire

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1991



ABSTRACT

This thesis presents criteria and necessary features by which to evaluate and design a good graphical user interface (GUI) for a Multimedia Database Management System (MDBMS). This material is also applicable to a traditional DBMS. Included in the thesis is the specification for a Query Management Facility (QMF) for a MDBMS user interface. The nature and benefits of the GUI environment, requires that we consider GUI concepts early in the user interface conceptualization and design. In today's DBMS user interfaces, these GUI concepts are for the most part applied as an after-thought. This is a critical mistake. Early incorporation of GUI capabilities along with established user interface principles results in a superior user interface. The QMF presented herein is one such interface. It combines the ideas of simple operations and data flow to allow the user to specify his query. Additional concepts used include: picture of the database schema, picture of the developing query, selectable objects, direct manipulation, piecemeal query specification, display of intermediate results and pre-defined joins. The resulting QMF is simple to use and enables the flexible expression of the simple as well as the complex database query.

1987
c.1

TABLE OF CONTENTS

I. INTRODUCTION.....1

A. BACKGROUND.....1

B. BRIEF DESCRIPTION OF OUR APPROACH.....2

C. SCOPE OF THESIS.....3

D. CHAPTER LAYOUT.....5

II. PREVIOUS AND RELATED WORK.....6

A. USER INTERFACE DESIGN.....6

B. GRAPHICAL USER INTERFACES.....8

C. GRAPHICAL USER INTERFACES FOR DATABASES.....9

III. RESEARCH ISSUES AND FINDINGS.....21

A. WHAT ARE THE CRITERIA FOR DETERMINING A GOOD DBMS GRAPHICAL USER INTERFACE ?.....22

1. Criterion 1 : The Proposed DBMS Graphical User Interface (GUI) Must Constitute an Improvement Over Existing DBMS Interfaces..... 22

2. Criterion 2 : The Proposed DBMS GUI Must Include the Integration of Applicable GUI Concepts and Capabilities.....23

3. Criterion 3 : The proposed DBMS GUI Must Support a Real-World to Database Mapping Mechanism.....25

4. Criterion 4 : The proposed DBMS GUI Must Support Flexible Expression of Query.....26

5. Criterion 5 : The proposed DBMS GUI Must Comply With Known User Interface Principles.....27

6. Criterion 6 : The Proposed DBMS GUI Must be Extensible.....27

B. WHAT ARE THE COMPONENTS OR FEATURES WHICH MUST BE INCLUDED IN A GOOD DBMS GRAPHICAL USER INTERFACE ?.....28

1. Provide a Simple Real World-to-Database Mapping..... 30

a. Feature 1 : Provide a Pictorial View of the Database Schema.....30

b. Feature 2 : Ensure the User can Easily Understand the Basic Building Blocks of his Database.....31

c. Feature 3 : Allow Visibility of Metadata.....33

d. Feature 4 : Allow Levels of Abstraction.....35

2. Maximize the Intelligent Use of Graphical Objects.....39

e. Feature 5 : Use of Selectable Objects.....39

f. Feature 6 : Use of Automatic Object Placement.....40

g. Feature 7 : Allow Easy User Selection and Arrangement of Objects.....41

h. Feature 8 : Use of Clearly Differentiable Objects.....42

3. Allow Stepwise Refinement of the Query During Formulation.....43

i. Feature 9 : Manipulation of Data Flow to Achieve Objective.....45

j. Feature 10 : Use of Simple Operations.....46

k. Feature 11 : Piecemeal Design and Construction of Query.....46

l. Feature 12 : Saving and Retrieval of Previously Defined and Commonly Used Joins.....47

m. Feature 13 : Immediate and Meaningful Feedback.....48

4. Minimize the Effort Required of the User.....49

n. Feature 14 : Ensure the Earliest Detection of Errors.....50

o. Feature 15 : Automatically Make Necessary Tools and Information Available.....51

p. Feature 16 : Stream-line Repetitive Actions.....53

q. Feature 17 : Use a Default Result Format.....54

C. APPLICABILITY TO MULTIMEDIA DATABASE SYSTEMS.....55

IV. DESCRIPTION OF GRAPHICAL QUERY MANAGEMENT FACILITY.....57

A. MAJOR FUNCTIONAL PARTS OF INTERFACE.....57

B. SIMPLE QUERY WITH INDIRECT USE OF MULTIMEDIA DATA.....59

C. SIMPLE QUERY WITH DIRECT USE OF MULTIMEDIA DATA.....65

D. COMPLEX QUERY.....68

E. AGGREGATE FUNCTIONS.....72

V. CONCLUSIONS.....114

A. APPLICABILITY OF APPROACH.....114

B. STRENGTHS AND LIMITATIONS OF APPROACH.....115

1. Strengths of Approach.....115

2. Limitations of Approach.....120

C. FUTURE WORK.....122

1. Implementation of the Proposed Query Management Facility.....122

2. Design and Implementation of Remaining User Interface Components for Database....123

3. Continuing Incorporation of Evolving Technology.....124

LIST OF REFERENCES.....125

INITIAL DISTRIBUTION LIST.....129

I. INTRODUCTION

This thesis is prepared in conjunction with the work of other researchers working within the area of multimedia database at the Computer Science Department of the Naval Postgraduate School. This thesis focuses on a subset of the issues one must consider when designing a graphical user interface for a multimedia database. The specific topic of interest is the query specification facility of such an interface.

A. BACKGROUND

There are some very good graphical user interfaces currently implemented to provide the quality of man-machine interfaces necessary to permit an optimization of the man-machine team. User interfaces of this quality do not exist for databases.

User interfaces for databases have received only a small amount of attention from the research community (ZODNIK90), (ENDU89). Many of the published papers present valuable ideas which are yet to appear anywhere but in a prototype form (ENDU89). There is a technology gap between the types of things database users are limited to as compared with the capabilities present in user interfaces for other areas.

Progress has been made in many areas. In the area of software progress has been made in graphics software, windowing software, networking software, operating systems and many others. In hardware progress has been made in screen capabilities, processing speeds, direct storage capabilities, hardware architectures and others. From the study of human factors engineering have come advances in user interface theory, guidelines and principles. The importance of database user interfaces must take advantage of all the technology and theory available and create the best user interface possible. The full impact of database technology can only be realized with the successful progress of user interface.

B. BRIEF DESCRIPTION OF OUR APPROACH

All database management systems have a user interface of some sort. The goal of this thesis is to come up with a user interface which is better, one which will also be applicable to a multimedia database. In order to do this the latest hardware and software technology is considered. Additionally sound user interface principles as well as a careful and creative consideration of user-to-database interaction within a multimedia environment.

The user interface approach taken in this thesis uses graphical direct manipulation as a means of facilitating the user-to-database interactions. The proposed user interface uses a unique approach to permit the user to express his

queries. The approach uses low-level data manipulation operations and the concept of data-flow while keeping the user close to the basic concepts of the relational data model. The approach integrates these ideas with each other. Additionally the proposed user interface integrates other user interface concepts and query specification ideas which have been found in related literature.

C. SCOPE OF THESIS

This thesis is concerned with the query specification process. In order to understand the scope of this thesis the reader must visualize a conceptual division of database management related functions. The functions of concern to us are :

1. schema definition/exploration
2. query specification
3. output display

Each of these functions is involved in a direct way with the query specification process. Terms have been coined to represent the portions of the user interface responsible for each of these functional areas. The schema definition/exploration is handled by a module called the Schema Management Facility (SMF). The query specification is handled by a module called the Query Management Facility (QMF). The output display is handled by a module called the

Report Management Fawility (RMF). These names were chosen for their descriptive value as well as their consistency.

As will become evident all three of these facilities are discussed and at least partially described in this thesis. This is not surprising considering the integral part they play in query specification. It should be remembered that the focus of this thesis is on query specification and hence only the QMF will be described in detail. Non-query related aspects of the SMF and the RMF will not be described.

The underlying assumption of the user interface presented in this thesis is that the database is one based on the relational data model. While many of the research findings presented in Chapter 3 are valid in the more general case, the proposed user interface presented in Chapter 4 is not. The decision to restrict the scope in this way is based on the usefulness and widespread acceptance of the relational data model. This decision is also effected by the fact that the prototype multimedia database at the core of the research teams efforts is based on the relational data model.

While this thesis is related to user interface research which is ultimately targeted for a prototype multimedia database, it has a broader applicability to a data base user interface for any relational database system. The multimedia aspects of the interface are for the most part found in the Report Management Facility which as stated above is not the central aim of this thesis.

D. CHAPTER LAYOUT

Chapter 2 discusses research conducted in the area of graphical user interfaces for databases as well as in related areas. Chapter 3 presents the research questions considered by this thesis. The chapter then follows with a description of the findings and conclusions resulting from studying and researching these questions. Chapter 4 presents a description of a proposed graphical user interface for a multimedia database. This proposed interface is based on the findings presented in chapter 3. Chapter 5 describes briefly describes future work required in the area of graphical user interfaces for multimedia databases. Chapter 6 provides a summary and conclusions reached about the proposed interface. This chapter discusses the strengths and limitations of the approach.

II. PREVIOUS AND RELATED WORK

This chapter groups previous and related works into three broad areas. The first section includes those works which relate to the broad area of user interface design for automated information systems. The second group presents those works which relate to the more specialized area of graphical user interfaces. The third and most interesting area includes those works which relate to graphical user interfaces for databases.

A. USER INTERFACE DESIGN

This section presents a couple of resources which are related to the design and development of any automated user interface. Interactive computer systems have been in use for some time. This implies an abundance of experience, research and literature. Much of the available literature is relevant to and of value when considering the design of a user interface such as that presented in this thesis.

Sidney Smith and Jane Mosier (SMITH86) have developed a collection of nine-hundred and forty-four guidelines for designing user interface software. This collection of guidelines includes general guidelines as well as very detailed guidelines. This work is mentioned just to provide support to the assertion that there are many guidelines and

principles published which one can consider when designing user interface software. With reference material such as this available there is no need of working design decisions in a vacuum. It is easy to obtain ideas and opinions when designing an interface.

Ben Schneiderman points out that prior to designing a system one should consider two import areas. One consideration is the nature of the user. The designer should acquire a good feel for the eventual user of the system (i.e. target audience). A second important consideration is the nature of the system and its tasks for which the interface is being built. (SHNEIDERMAN87)

Schneiderman offers a set of eight principles which apply to user interface design. These principles provide a good checklist for ensuring that a design has at least included consideration of issues commonly applicable to user interfaces. These principles, presented in Table 1, are general in nature but as a whole they capture most of what different authors have said in this area. These eight principles will be considered further after the proposed DBMS user interface is presented. They will be used during part of the post-evaluation of the proposed interface.

TABLE 1 - USER INTERFACE PRINCIPLES (SHNEIDERMAN87)

1. Strive for Consistency.
2. Enable frequent users to use shortcuts.
3. Offer informative feedback.
4. Design dialogues to yield closure.
5. Offer simple error handling.
6. Permit easy reversal of actions.
7. Support Internal Locus of Control.
8. Reduce short-term memory load.

B. GRAPHICAL USER INTERFACES

Dan Heller discusses issues related to the use of currently available graphical user interface tools. He speaks about the high level tools used to develop user interfaces in today's windowing environments. Heller claims that all of the GUI tools are based on some already accepted fundamental user interface principles (HELLER90a). This statement suggests that by using standard GUI tools, compliance with fundamental user interface principles is maintained. The built-in enforcement of sound user interface principles in this case is a beneficial side-effect of using a GUI. Heller (HELLER90a) presents a list of graphical user interface design goals as presented in Table 2. This list of goals was specifically developed in relation to GUI's which tailors to the types of concerns relevant to the GUI developed within this thesis. The goals listed in the table will be used for the design and evaluation of the proposed DBMS user interface.

TABLE 2 - IMPORTANT GOALS IN GUI DESIGN (HELLER90a)

1. **Intuitive**
2. **Consistent**
3. **Conducive to Frequent Use**
4. **Visual Cues**
5. **Flexible**

C. GRAPHICAL USER INTERFACES FOR DATABASES

With a graphical user interface, a data model must be presented to the user. This data model is necessary to facilitate his interaction with and understanding of his database. Different data models and ideas about how to use them for browsing, querying and working with a database have been studied. A consistent theme presented is the clarity with which the model must allow the user to understand his database. Different models have been proposed. The entity-relationship (E-R) model has been a central part of several proposals (ELMASRI85), (PAOLO83), (ROGERS88), (WONG82), (ZHANG83). This choice of the entity-relationship model is a good choice and hence has also been adopted by the user interface presented herein. These related works will each be discussed below. Elmasri and Larson (ELMASRI85) present a sequential query specification process whereby the user begins his query by first specifying the entities and relationships of interest. This is facilitated by a pictorial use of the E/R model. Next the user causes the entities to be automatically arranged into a hierarchy by manually selecting one of the entities as the root entity. This assignment automatically

implies the structure of the remaining hierarchy since the relationships of the initial E/R diagram are assumed to hold. Next the user places selection conditions on first the root entity and then the remaining entities. Based on where the user is in the query, his conditions will have different and difficult to foresee results. To overcome this the system interprets the user specified condition and comes back with a display of its own interpretation. The user then states whether the system interpretation is what he intended with a Yes or No. A No response from the user causes iteration of the process. This specifying of conditions on entities is in effect causing a selection on each relation. The user next selects the desired attributes from each entity. These last two steps can be reversed or interleaved. (ELMASRI85)

Selecting entities and relationships from an E-R diagram is similar to our approach. Also the notion of pictorially seeing the structure of the query by a graphical representation of entities, or in our case, relations. The remainder of the process described by Elmasri and Larson is not considered useful (ELMASRI85). It should not be expected that the user can define all the entities and relations at the beginning of a complex query. This places an unacceptable burden on the user. Additionally the user should not be forced to conceptualize his query in the form of a hierarchy with his identification of the root. This certainly is not intuitive. In contrast to this our proposal uses the idea of data flow

from the leaves to the root of an upside-down binary tree. The leaves or initial relations can be specified at any point during the query. The root is the final relation which represents the result of the query. Our approach allows the user to think as he goes and to change his mind on the fly.

When specifying conditions in a complex query the user should not be expected to use a syawem interpretation of his condition for an iterative dialogue. In a sense this is like having the user throw darts until he gets a bullseye. This does not put the user at the locus of control. In contrast our proposal provides meaningful feedback to the user at each application of an operation. There is no complicated data structure for the user to consider before applying an operation. There is simply one or two relations which are input to an operation. There is only one relation which is an output to an operation. The user may then immediately see metadata or actual data represented by this relation. If the operation is not what he intended he can go back and retry this single operation. There is no guessing of what the system is doing. The system responds to each of the users actions and immediately supplies the user with helpful feedback.

Rogers and Cattell (ROGERS88) present an implementation of a database user interface which facilitates schema design and database browsing. The facility called "Schemadesign" permits the user to define a database while pictorially providing a bit-mapped graphic display of the developing E-R semantic

model. This definition process encourages data normalization, appropriate definition of keys and declaration of applicable integrity constraints. The backend is a relational database which assumes responsibility for supporting the maintenance of declared integrity constraints. A second facility called "Databrowse" is a window-based program which allows the user to browse the schema and edit values in the database. The browsing and displaying of data is centralized around the concept of "entity". This means the user is at times removed one layer above the concept of a "relation". The display of a record for an entity may contain data from many relations with many records from some of the relations. This display of many records for viewing one entity results from the nature of multi-valued attributes of a relation. Databrowse uses a degwult form to display data about entities. The display and its underlying database support the handling of certain types of unformatted data. The system uses a binary data type for storage of unformatted data within the database as well as having a means to store pointers to files residing external to the core back-end database. The level of support provided for unformatted data is not described in a detailed fashion. (ROGERS88)

The Schemadesign program described by Rogers and Cattell is considered partially useful as a basis for integration into the prototype interface proposed herein. This is based on initial considerations and is not final at this point. As

stated previously the focus of this thesis is to present a Query Management Facility (QMF). The interface presented in this thesis does not include a well defined description for a Schema Management Facility (SMF). It only presents those considerations which must be considered. Key amongst these are that the SMF must be based on the E-R model, capable of supporting the relational data model and be graphical as well as to permit graphical manipulation. Additionally the SMF must appear functionally consistent and interactively similar with the QMF (e.g., the user can not be expected to learn two significantly different systems for working with one database). On the surface it appears that the Schemadesign program described by Rogers and Cattell meets these guidelines. The drawback in the Schemadesign program is its focus on entities vice relations. This adds an unnecessary level of confusion to a user of a relational database. During a query the user must focus on relations and be aware of all the relations which might be a part of his query. This includes relations representing multi-valued attributes of entities. The QMF proposed in this thesis has the user working with such relations the same as any other relations. This is for the reasons of consistency and simplicity. The relation is the common denominator for all data used in queries. The SMF must support this same approach. (ROGERS88)

The Databrowse program mentioned in the work by Rogers and Cattell is useful in casual browsing of the database. It does

not appear to support the expressiveness needed by a query facility. It is not suitable as the query tool sought in this thesis. The authors do mention intentions to develop a more sophisticated query tool in the future which will be designed to use graphics. The design decision made in the design of Databrowse to use default forms and to work in a windowing environment are consistent with the decisions presented in this thesis. Databrowse works on the concept of the entity vice the concept of relation. This is in contrast to our proposal as explained in the paragraph above. (ROGERS88)

Wong and Kuo (WONG82) present a user interface called GUIDE (i.e., Graphical User Interface for Database Exploration). GUIDE is a graphical user interface for a database which is based on a network of entity and relationship types. The E-R model is pictorially displayed for the user and is an integral part of the query expression process. Queries are expressed as traversal paths on the E-R network.

GUIDE contains several very important features which are included in the user interface proposed in this thesis. GUIDE permits the user to formulate his query in a piecemeal fashion. As the user is piecing his query together he can look at intermediate results. These features encourage database exploration and constitute meaningful feedback. GUIDE contains a mechanism to allow the user to define varying levels of detail in the presentation of the schema. It does

this by allowing the user to set the level of detail (e.g., on a scale from one to five) and also to set the radius within which objects are to be displayed. This radius is based on the number of links traversed from a central object which is user determined. GUIDE also allows the user to toggle off the display of specific unneeded objects. Additionally GUIDE enables the user to acquire detailed information or explanation on a user specified object. This information includes the display of available meta-data. These features amount to allowing the user to deal with the desired level of abstraction and to hide information which is not essential to his user objectives. These features are all considered an important part of a graphical user interface for a database.

There are a number of things which set the interface proposed in this thesis apart from GUIDE. One difference stems from GUIDE's presentation of the database schema and display of the developing query both in the same picture. This over-tasks the differentiable symbology and overworks the screen area responsible for conveying information to the user. Colorization of objects is used to show the user the relevant parts (i.e., entities and relationships) of his developing query. It appears that as pieces of a complex query begin to overlap, the visibility of local queries as well as the overall query are lost. Additionally as pieces of a query are linked together, GUIDE appears to automatically build these links. This takes the necessary feeling of control away from

the user. From the description of GUIDE, there is no apparent means to resolve decisions when more than one linking alternative exists.

GUIDE buffers the user from ideas tied solely to the relational data model. The authors claim that concepts such as the relational join cause the user too much memorization and effort in explicitly specifying implied relationships.

In contrast to GUIDE, the interface proposed in this thesis uses separate pictures to display the schema and the developing query. At all times during the query formulation it is made clear to the user where he is at and how the pieces of the query fit and interact together. To a large extent this is achieved through the use of data flow, simple operations, and consistent meaning of displayed objects. The user is always made to feel in control as everything occurring during the query formulation is caused by him.

The proposed interface recognizes the additional effort and memorization which may result when requiring the user to use low level relational operations. This factor is mitigated by allowing the user to use the E-R diagram to specify implied relationships. The use of pre-defined joins and previously defined queries is believed to more than offset the cost associated with low-level operations. Additionally, low-level operations permit the user to acquire an increased understanding and appreciation for the capabilities of his

database. This is invaluable when confronted with the formulation of a complex query.

Zhang and Mendelson (ZHANG83) propose a graphics based entity-relationship query system. In their proposal the user is presented with a picture of a database schema on a graphics screen. Queries are formulated by pointing at nodes to be included in the query, using a mouse. Conditions on the nodes are specified by filling in forms. These forms are QBE-like forms completed by typing in expressions constructed of constants and variables. Three operations which can be performed on relations, (i.e. union, intersection and difference) are presented as menu options. These operations are used to combine the results of several elementary queries. They claim navigation is simplified in two ways, one being the display of the diagram on the screen and the other is the use of default connection paths. A default connection path is a minimal connected sub-graph connecting the specified nodes. These default connection paths are computed by the system based on the notion of maximal objects (MAIER83). At any time during the query formulation the user may request execution of the query as specified and continue refining the query after viewing the results.

Zhang and Mendelson present several very key ideas. One of these is the notion of the incremental query formulation. This allows the user to perform the query in a piecemeal fashion. One restriction which appears to exist in their proposal is

the arbitrary combination of the relational operations. It appears that selections, projections and joins constitute an elementary query. Only after two or more elementary queries are formed can the union, intersection, or difference operations be applied. Once these operations are performed it appears as though no further query refinement is permitted. Our proposal overcomes this problem by maintaining that all intermediate results are relations. By doing this it is ensured that all operations can be performed in any order and at any level of complexity. This is of course under the condition that the user chosen operation makes sense on the relation or relations chosen as operands.(ZHANG83)

Another problem present in the proposal by Zhang and Mendelson is the non-uniformity in presentation to the user. Some operations may only be applied through QBE-like forms, while other operations may only be applied through menu selection. Queries by their nature may be inherently complex. As the user formulates the query through the user interface of a DBMS, a simple means of expressing the complex query must be provided. The user must be able to consistently apply database operations on data. (ZHANG83) Our proposal ensures this is satisfied. All operations are applied in the same manner. The may be applied at any time and conceptually result in a consistent action (i.e. modification of the data flow). More will be said about this in the next chapter.

A well known database user interface is Query By Example which is also known as QBE (ZLOOF75). With QBE the user is presented with a tabular view of the database. The user begins a query by selecting a subset of all the tables in the database. The expressions needed to select the desired rows of data and attributes to be disawayed are then specified with respect to each table. The tabular format is simple to understand but fails to communicate some of the semantics of the data. When data from more than one table is needed, the user must remember what attributes from what tables result in the desired joins. There is no notion of default joins as modeled in the E/R model. There is also no means for identifying predefined joins. Overall the QBE approach is not considered helpful in our quest for a good database user interface.

The use of forms for user interfaces is not new (WARTIK88). An interesting forms type interface for databases is called FOSTER (MIYA087). This is a forms-oriented user interface using forms which are very close in format to those used in QBE. With this interface the person performing a query uses graphic icons, directed edges and forms to create application programs. These application programs can then be used to query the database. The problem with this approach is complexity in defining forms and integrating them into the iconic language. The entire approach requires a lot of work and learning by the user. It appears that the user would have

to spend time to become quite familiar with a users manual in order to figure out how to begin. In contrast, the user interface presented in this thesis is quite simple. With a small amount of learning the novice user can use the interface to perform difficult queries in a simple manner. Information and tools (e.g., menus) which are needed by the user are automatically made available to the user at the appropriate time. Guidance is provided to the user in the form of suggested actions. As the user is performing actions a dialogue takes place guiding the user through his work. If he for example selects a binary operation, the interface will ensure he chooses two appropriate operands.

Other less known models such as the Graph-Oriented Object Database Model (GOOD) (GYSSENS90), User Data Model (MIYAO86), and Semantic Data Model (SDM) (GOLDMAN85) have also been proposed. A model such as GOOD does not lend itself to concepts of the relational model, but is more suited to use with the object-oriented data model. The User Data Model is not considered helpful in presenting the user a clear view of the schema. The SDM does not lend itself to the low-level query specification approach proposed in this thesis. These models are mentioned to highlight the fact that other models have been looked at but the E-R model seems to be the best available semantic data model for our purposes while working within the realm of a relational database.

III. RESEARCH ISSUES AND FINDINGS

This thesis addresses two basic research questions. In this chapter the issues and findings related to these two research questions are presented. The two questions are :

1. What are the criteria for determining a good DBMS graphical user interface ?
2. What are the components or features which must be included in a good DBMS graphical user interface ?

The first question relates to the establishing of a benchmark or yardstick by which to gauge the relative quality of a DBMS user interface. Neither a benchmark nor a set of standards by which to gauge a DBMS user interface exists. This question addresses the things which one might consider as part of such a benchmark. It is an important question since if we do not have a set of such criteria then we have no way of evaluating and thereby progressing forward in this endeavor. We are in a broad sense laying the groundwork for future advancements in the area of DBMS user interfaces. This groundwork is equally applicable to the more specialized area of multimedia database systems.

The second question attempts to capture in a detailed way the types of features which must be included in the design and development of a graphical DBMS user interface. As with the criteria mentioned above, these features include those which

are necessary for a user interface for a multimedia database. As each of the features are described, a discussion and one or more examples from the proposed DBMS GUI are presented.

The material corresponding to each of these two questions is presented below, under the respective sub-heading.

A. WHAT ARE THE CRITERIA FOR DETERMINING A GOOD DBMS GRAPHICAL USER INTERFACE ?

1. Criterion 1 : The Proposed DBMS Graphical User Interface (GUI) Must Constitute an Improvement Over Existing DBMS Interfaces

This is not to say that current DBMS user interfaces are not "good", in the general sense of good. Current database user interfaces do work. The assertion made by this thesis is there exists a graphical user interface for databases which is better than those currently implemented. This is what we search for. Only when such an improvement is found, is it considered good. To know when this objective has been reached there must be a comparison against existing database user interfaces.

There are a number of different methods currently implemented or proposed in literature to query a database. These include linear query languages such as SQL and two-dimensional methods such as QBE. One means of evaluating a new proposal is to compare it against these other methods which are similar in function and ask, "Is this an improvement ?" If

the proposed query interface is not an improvement then for our purposes we can not classify it as a good interface. The bottom line is we are looking for something which is measurably better.

What does "measurably better" mean ? The problem we encounter is there is not a standard way to measure such a qualitative thing as a 'better interface' (WU89). This is in part responsible for the lack of attention given this area by computer scientists. Enhanced productivity, enjoyment by the user, amount of learning required and number of mistakes made by the user are some of the factors which should be considered in this situation. This is a criterion which is best studied within the domain of human factors engineers. For our purposes we will not consider the scientific techniques employed by such engineers. We will simply rely on persuasive argument based on subjective intuition to claim we have an interface which is an improvement. Still, the idea is valid that there must exist a reasonable assurance that a new proposal is in some sense an improvement in order to consider it a good DBMS graphical user interface.

2. Criterion 2 : The Proposed DBMS GUI Must Include the Integration of Applicable GUI Concepts and Capabilities

Another important criterion for evaluating a DBMS GUI is the degree to which it integrates the latest GUI concepts and capabilities. A good DBMS GUI must include integration of those graphical user interface capabilities which through

their integration, make it a better interface. GUI concepts should not be included just for the sake of inclusion. Each concept should be carefully considered. It should be considered in conjunction with other user interface tools and concepts. The ultimate goal is to achieve the best overall interface through a skillful combination of existing GUI tools. Consistency for the user is an important goal to keep in mind when integrating various tools (WARTIK86). It may be the case that certain concepts and capabilities should not be used. This case should occur only through a conscious design decision. (ANDERSON86)

Two examples of such GUI capabilities are those involving windowing tools and those involving Graphical Direct Manipulation. Windowing tools includes the software which enables such things as the creation, movement and re-sizing of multiple windows on a terminal screen. Numerous ways of applying windowing capabilities have been either implemented or proposed. Graphical direct manipulation refers to the capability to manipulate and use visual objects through the use of a pointing device such as a mouse (KUNTZ89, SHU89). This includes operations such as moving, re-sizing and pointing at objects for selection purposes. A wealth of related concepts as well as different ideas have either already been implemented or have been proposed in research (KUNTZ89, BRYCE86). All of the applicable concepts must be

considered to ensure that an interface does not exclude something which would make it better.

3. Criterion 3 : The proposed DBMS GUI Must Support A Real-World to Database Mapping Mechanism

This criterion refers to the effectiveness with which a user interface allows the user to understand what is in his database. The user presumably knows about the real world or at least about that portion which is modeled in the database. The user must clearly understand what is and what is not modeled in his database. The user interface must not assume the user knows the detailed contents of the database (ELMASRI85, pg 237). To make good use of the data the user has got to have an appreciation for what the database does and does not contain as well as the types of queries that it can and cannot support. A powerful data model which captures all sorts of meaningful information does not do any good if the user cannot request and subsequently extract the useful information. The user interface is what lies between the user and the data. The user interface must allow the user to see into his database in a way which is intuitive while at the same time map this data to the users perception of his real-world. Another way to say this is that the user interface must include a good mapping mechanism which allows the user to understand the relationship between the database and the real-world (WONG82).

4. Criterion 4 : The proposed DBMS GUI Must Support Flexible Expression of Query

The user of a DBMS may require a database to respond to an extremely complex query. To be complete the database and it's user interface must support the expression of and performance of such a query (ELMASRI85, pg 238). The expressiveness of a query language is difficult to gauge. There is the notion of relational completeness. This refers to the expressive power of a query language and represents the minimum capability of any reasonable query language for the relational data model (ULLMAN82, CODD72). A database based on the relational data model must at a minimum be relationally complete. There are a class of queries known as recursive queries. A DBMS user interface must support recursive queries. There are also a number of aggregate functions which provide a convenient tool for the database user (e.g., count, sum, avg, min, max). A good user interface must include the provision of the basic aggregate functions.

There exists a deeper notion of what a database query expression capability might support. Consider the case of a single complex query. Assume three different users have the need for the results of this query. There are a number of ways to think about this query. It is reasonable to expect different users to think about what is in essence the same query, in different ways (ELMASRI85,pg 237). In this example, assume that each user thinks about the query differently. A

database query specification facility should allow a user to express a query in a way which he thinks about the query. The user should not have to re-think his query in order to express it in a way which the system will understand. In the example, each of the three users should be able to express the query in a way which is intuitive to him. This implies that the user interface should support whatever formulation of the query the user might naturally happen to express. This is considered as flexible support of query expressions. A good DBMS user interface must support flexible expression of queries.

5. Criterion 5 : The proposed DBMS GUI Must Comply With Known User Interface Principles

User interfaces in general sense have been around for a long time. Over this period a substantial amount of research and number of studies has gone into this area. As a result a number of guidelines and principles have been proposed. Some of these guidelines and principles are relevant to the user interface we consider in this thesis. A summary of some of the more important principles and guidelines are mentioned in the Section A of Chapter 2. A good DBMS GUI must comply with these basic principles and guidelines.

6. Criterion 6 : The Proposed DBMS GUI Must be Extensible

What constitutes a good GUI today may not constitute a good GUI tomorrow. Things continually change and as they do, new user interface and database concepts continue to be

discovered. In addition to this new hardware capabilities, and new software capabilities continue to evolve. As these changes occur a good GUI will grow and change with them. For this reason, a DBMS GUI must be designed and built to grow and change. This flexibility can only be ensured through the use of a modular design. A user interface must be designed into functional modules with well-defined interfaces amongst the modules. There must also be well-defined interfaces between the user interface modules and the underlying database. Not only will this enhance software supportability but it will also provide the extensibility required.

It is clear that the user is not directly aware of the qualities of a user interface which make it extensible. This criterion is one which supports those aspects of a user interface of concern to those in the business of designing, implementing and maintaining user interfaces. The implication here is a good user interface lends itself by way of its design to future evolution and maintenance.

B. WHAT ARE THE COMPONENTS OR FEATURES WHICH MUST BE INCLUDED IN A GOOD DBMS GRAPHICAL USER INTERFACE ?

As previously stated the focus of this thesis is on the user interface aspects of database query formulation. Chapter One mentions three related functional facilities of the DBMS user interface. The Query Management Facility is the key facility being considered. The other two functional facilities

which play a key but supplemental role in query specification are the Schema Management Facility and the Report Management Facility. This structural overview is repeated here to aid the reader in understanding the framework in which the following material is presented.

This section of the chapter describes the necessary components of a good DBMS graphical user interface. The required components are presented as a detailed list of features which should be a part of a good interface. This detailed list is presented below. The features have been arbitrarily placed into one of four groups based on the primary benefit to be derived the feature. These four groups are :

1. Provide a Simple Real World-to-Database Mapping
2. Maximize the Intelligent Use of Graphical Objects
3. Allow Stepwise Refinement of the Query During Formulation
4. Minimize the Effort Required of the User

This does not mean that clean lines exist with which to differentiate these features. It only means that this is the arbitrary grouping that I have chosen to use. There is significant overlap in benefit to be derived from many of the features.

As each feature is presented below, one or more implementation examples will be discussed. This example or

examples will relate each feature with one or more of three major functional facilities of a DBMS user interface which were mentioned above (i.e. , Query Management Facility, Schema Management Facility, Report Management Facility).

1. Provide a Simple Real World-to-Database Mapping

This group includes features which enable the user to easily understand his database. The assertion which belies the entire query specification approach presented in this thesis is the user can and should be able to easily understand his database. The ease and effectiveness of this understanding, depends on how good the mechanism is which allows him to make the intellectual mapping between the real world and the model of the real world which is represented in his database.

a. Feature 1 : Provide a Pictorial View of the Database Schema

The Schema Management Facility (SMF) must provide a pictorial view of the database. It is assumed here that the database is structured in accordance with the relational data model (RDM). In this case the choice is either to directly present the user a picture of the database structure in a format representative of the Relational Data Model or to use an intermediate model which is closer to the users perception of the world which he understands. The E-R model sufficiently provides this intermediate model (WONG82).

The idea is that the user must be able to visually relate the concrete aspects of his world to the abstract

representation in his database. A picture allows the quick and simple communication (i.e., mapping) of the database model for the user. The choice of the E-R model here is arguable and other models have been proposed (MIYA086, MIYA087, GYSSENS90). A comparative evaluation of the different models is not within the scope of the thesis and will not be further discussed. The interested reader might desire to consider the proposal presented in this thesis with the assumption of some other conceptual model. The point is a pictorial (i.e., visual) representation is essential. As will become evident later, this picture will facilitate important user activities such as selection of desired objects within the database. In light of the assumption we have made about the availability of a high-resolution color screen, a pictorial means of communicating the structural and semantic state of the database is unquestionably a necessary and supportable feature.

b. Feature 2 : Ensure the User can easily Understand the Basic Building Blocks of his Database

In order for the user to achieve a high level of comfort with and comprehension of his database, he must feel comfortable with its basic building blocks. This is also critical if he is to effectively comprehend a complete mapping from the real world to his database. There must not be gaps or empty holes where things mysteriously happen. As far as the user is concerned, the relation or table is the lowest level in his database. It is simply a grouping of related data about

a given entity or relationship. The table is organized into rows and columns. The user can and should be made to feel comfortable with this concept. The relation is the basic building block of his database. It is also the initial, the intermediate and the final result during the query specification process. The user is always selecting and performing operations on relations. There is a relation going into each operation and a relation coming out of each operation. It is the user's goal to complete his query by creating the table (i.e., relation) containing the desired data.

The Query Management Facility (QMF) must allow the user to select objects with which he may work and also to manipulate objects through the application of various operations. Through graphical direct manipulation (GRDM) the operations on objects are made easy for the user (KUNTZ89). More will be said about GRDM later. The object with which the user works during query formulation should be pictorial in nature and the meaning must be something the user can easily map to the real world. After all it is something about the real world that the user is trying to get as a result of his query. The pictorial objects used consistently through the QMF are relations (i.e. tables). At every step throughout the process of query formulation, the input as well as the result of the query operation is a relation. These relations are consistently displayed as rectangles. The user may think of

them as tables of data. In his mind he must manipulate these tables of data to achieve his desired goal. The idea that a table contains columns and rows of data is not a difficult concept for the user (ULLMAN82,pg.168). A user can easily handle the concept that his data is store in such tables. The users ability to understand this idea has been argued in the literature (MIYAO86) but the feeling here is that sufficient credit must be given to even the naive user for being able to deal with this simple concept.

c. Feature 3 : Allow Visibility of Metadata

An assertion made in this thesis is that the user can and should understand his database. This requirement is supported in part by Feature 1. Feature 1 provides a pictorial display of the database schema. An additional means of supporting the users understanding of the database is through the availability of textual metadata. In simple terms metadata is data about data. In this case, metadata includes such information as the number of tuples in a relation. Other metadata helpful to the user includes the number of attributes in a relation and attribute type and size information. This type of information helps provide the user with a warm feeling about the correctness of what he is doing during the query formulation process. It provides a sort of feedback about the physical results of various operations performed by the user during the query formulation process.

The value of metadata can best be explained through an example. Consider for this example, the case of a company database. A user generally has a good feel for the nature of his data. This follows from the idea the database reflects the users real-world. For example, assume that a personnel clerk knows that he has in the vicinity of 200 people in Department A and about 2000 people in his entire organization. He also knows that about 20% of the employees are salaried employees. If he is querying his database for a list of the people who have are "Salaried Employees in Department A", then he has some feel for the numbers involved. As he starts with the employee relation he may see metadata indicating that there are 2138 rows in the relation. This number makes sense to the clerk. After performing a selection for those employees in Department A who are salaried he observes that the resulting relation has 37 rows. Without much thought the personnel clerk now has some intuitive feel for the correctness of this portion of his query. (i.e., He expected to see a resulting relation with somewhere between 20 and 50 rows). This is an example of the value which is provided by meaningful feedback to the user.

An additional source of metadata is that data one normally might find in the data dictionary. This is information which is specified during the time of the database schema definition. This data includes such things as sponsors of certain attributes, or tables within the database. It may

also have comments explaining the practical meaning of attributes or tables or the relationship between them. This type of information is made available to the user in the proposed interface.

Metadata is meaningful feedback. The proposed interface allows the user to view this type of feedback as he is working with the data. All of this supports the user feeling comfortable about his perception of the correspondence between the real world and his database.

d. Feature 4 : Allow Levels of Abstraction

Levels of abstraction is a key concept which must be generously integrated into any good database user interface (LEONG89). We as humans have a very limited ability to deal with large quantities of information. Even with this fact we have found ways to accomplish enormously complex tasks. We do this through our innate ability to abstract from that information which is available, the information which is relevant to what we are doing.

This concept is applied throughout the proposed interface. Whenever possible, the user is permitted to see more detail or less detail as he requires. This is intended to assist him dealing with the simple and more importantly the complex queries which he may be required to perform. Some applications of the levels of abstraction feature are mentioned below.

Through the Schema Management Facility (SMF) the user is presented with an E-R diagram of the database schema. In this diagram there is a means for the user to cause the display or non-display of the attributes. The user may also cause the inclusion or exclusion of the relationship cardinalities in the diagram.

As mentioned in Feature 3, the user may view the metadata as he is formulating his query. The interface permits the user to turn the display of this data on and off. As the user is formulating his query, he is creating and working with its' graphic representation. The user may view metadata as cluttering up the picture. He may choose to see this detail only when he is uncertain about the correctness of a particular step. In this case, it is open to the user to choose the desired level of abstraction.

This thesis assumes no specific level of sophistication on the underlying relational database management system. The user interface will however give the user ready access to see whatever data the underlying database maintains about the database. This includes the data which one might find in the Data Dictionary of a large DBMS. Such things as the owner of certain data, the long names of the attributes, the aliases of the attributes, English descriptions of the attributes and whatever else is available. Similarly this claim applies to data maintained on the relations comprising the database. Clearly this could amount

to a lot of information and should be available to the user only on an on-call basis vice as a default. In this case, too much information could be as bad as not enough.

The database underlying the user interface might be quite large and complex. It might be used by a number of different users each with his own interest in the database. Even a single user might have special parts of the database which are commonly used and others which are seldom used. For this reason and as well as to support the other benefits of abstraction, the capability to create customized views of the database is required. This means, that as the user is viewing the database schema, he is seeing the portion of the database of concern to him. This idea ripples into the other pop-up windows and so forth which provide quick selection menus to permit selection of desired table names, field names, etc. These all automatically remain consistent with the users customized view of the database schema. The assumption is that the user has put those items of interest into his customized view of the schema. Since this functionality is for the most part a product of the Schema Management Facility (SMF) the implementation specifics will not be laid out within this thesis. Suffice it to say that its inclusion in a good SMF is important and we assume its presence.

Another feature which must be included in both the Schema Management Facility (SMF) and the Query Management Facility (QMF) is the ability to zoom in and zoom out on the

picture currently being displayed (WONG82). This becomes especially important when the user is working with a large database schema and is incrementally formulating a large multi-step query. The surface of the terminal screen is only so large, therefore a zoom option must be used to allow the user to step back and get the big picture yet still be able to zoom in to work with the necessary level of detail to accomplish his task.

Another example of the concept of levels of abstraction is the capability to permit implosion and explosion of local queries and previously saved fragments to show or hide levels of detail. As the user is incrementally building a query he is creating building blocks which will be a part of his final query. These are called local queries. These same blocks might also be saved in their present form to be used later in a different query. In a later query, these would be called previously saved fragments. In its most detailed form a previously saved fragment or a local query might consist of a number of relations and edges between these relations as well as graphic icons representing operations. To facilitate ease of working with these building blocks, they may be aggregated into a simple block representing the net relation, which can then be exploded later if the user desires to see or work with the inner detail.

2. Maximize the Intelligent Use of Graphical Objects

This thesis explores database user interfaces in light of Graphical User Interfaces (GUI's). This is different from the traditional textual displays with textual menus where the user either types a 1, 2, or 3 corresponding with his menu choice or types in English text in response to system prompts. A GUI makes use of graphical objects. A graphical object is anything displayed on the terminal screen. This includes different types of windows, scroll bars, buttons, pictures, icons, etc. The simple display of text data is not a graphical object. The use of the capabilities uniquely available with GUI's should be used to the maximum extent possible. One should bear in mind, however, that the application of these capabilities must be used in an intelligent manner. Some features which apply this idea are presented below. (KUNTZ89)

e. Feature 5 : Use of Selectable Objects

While formulating a query the user must make a number of choices. He must for example decide which relations he wants to use in the query. He must also decide which attributes he wishes to include during a selection of the columns to include. This type of choice can be accomplished through pointing and clicking on a textual description of the items on a menu presented in a pop up window. This is not a bad approach and is an option the user has in the proposed interface. There is another method which should be given to the user of a DBMS GUI. As mentioned previously the use of a

picture of the schema is a necessary part of the interface. A picture of the developing query is also available to the user. These pictures are something which the interface encourages the user to look at. The user just through using the interface understands what is going on in the pictures. This provides the key reason the pictures (ie graphical objects) must provide a medium through which the user can communicate as he develops the query. If the user wants the EMPLOYEE relation then the user should only need to point to the EMPLOYEE object and click. If the user wants to include the NAME and ADDRESS attributes the user must only point to them and click. This sort of capability to point at selectable objects must be part of a good GUI. It provides significant savings in time and effort over navigating and reading textual menus.

f. Feature 6 : Use of Automatic Object Placement

When graphical objects must be placed into a work area they should be automatically placed wherever feasible. This saves the user the time it would take to drag the item to a position and drop it. A good example of this is included in the proposed interface. While using the Query Management Facility (QMF), the user points to and selects one or more objects (i.e. relations) to serve as operands. He then points to and selects the desired operation. The visual result is the creation of a rectangle representing the new resulting relation. As a rule, the desirable placement of this new object is generally below the operand and centered. If the

user were placing the item he would almost always place it in accordance with this rule. For this reason the interface should do it for him. This saves the user the work necessary to do this placement. To allow for the exceptions when the user desires a different placement of objects, the interface provides an easy to use ability to move objects. This capability is discussed as the next feature presented below.

g. Feature 7 : Allow Easy User Selection and Arrangement of Objects

The interface must provide the user an easy means to work with graphical objects. This is especially important in an environment such as the proposed interface where so much of the users attention is focused on the visual objects. Here we are not talking about moving windows. As mentioned before we presume that this activity is within the domain of a standard window manager. This feature refers to the objects presented in the SMF and the QMF. The QMF is providing the user with a current picture of the state of his query. This picture should be arranged in a way amenable to the user. If not, the user must be able to easily rearrange it. This is possible by allowing the user to select an object, drag it and the drop it in its new location. Another important point involves selection of multiple objects. Presume that the user desires to save a piece of a large query he is working on, The piece involves twelve objects (i.e. rectangles) on the QMF work surface. The user must only drag the pointer around these

objects and then release the button to select the group of objects. This is much easier than pointing at and clicking on twelve separate objects.

h. Feature 8 : Use of Clearly Differentiable Objects

A large amount of the user's attention is drawn to the pictorial view of the database via the SMF and to the pictorial view of the developing query. For this reason it must be very easy for the user to remain clear about what he is looking at. Consider the picture of the query. The user is looking at relations and at the flow of data between these relations. It is easy for the user to realize that rectangles mean relations and lines mean data flow. These are two very different concepts represented using two very different symbols.

There is a subtle difference amongst relations. Some relations may represent actual relations in the database (i.e. base relations) while others may be virtual relations in the sense that they are derived via the application of operations from base relations. This is a difference which will rarely make a difference for the user but still the difference is important. To represent this difference, the base relations are presented in a different color from virtual relations.

The user is frequently selecting objects during the process of formulating his query. When objects are selected, this special status of the object must be clearly

discernable to the user. The user interface uses changes in the intensity and texture of displayed objects to ensure this occurs.

3. Allow Stepwise Refinement of the Query During Formulation

Each time the user approaches the database, he has an objective to accomplish. The DBMS user interface must make the accomplishment of this objective as easy as possible. The users objective falls into one of three categories. First it may be to conduct a simple query. In this case the interface must provide him a means to perform the query in a simple and efficient way. The second objective the user may have when approaching the database is the formulation of a complex query. The third and final objective the user may have is to explore the database. These final two objectives are both facilitated by permitting the user to perform his work incrementally and to iterate over certain steps when necessary (ELMASRI85, KUNTZ89, LEONG89). Both the incremental formulation of the query and the capability to easily iterate over a step are discussed below.

The incremental formulation of a query is essential in enabling the user to successfully communicate and formulate a complex query (KUNTZ89, LEONG89). Incremental query formulation as an approach, is consistent with the way people think when confronted with complex problems. There is a correlation between the database users need to communicate and

formulate a complex query with the more general notion of peoples need to solve complex and unfamiliar problems.

When faced with solving complex problems, people naturally tend to break the problems into smaller, more manageable parts. This approach is quite natural and quite common. By considering the smaller parts of a problem a person can overcome the general ambiguity which comes with a problem too complex to consider as a whole. Once the person comes to understand the component parts of a problem, he can begin to consider the relationship between these parts. Once these relationships are understood, the nature and comprehension of the larger problem comes into focus. In the best case the person is successful in understanding all the component parts of a problem as well as all the relationships. This leads to a complete understanding of the overall complex problem. In the less then ideal case, the person understands only a subset of the component parts of the problem. This case still provides a person with a better understanding then he started with.

Incremental query formulation makes use of this approach of dividing and conquering complex problems. Assume that the user has a complex query which is not easy to communicate. The user can specify the parts of his query which he does understand. He can then establish the relationship between these parts. Features 9 through 13 support incremental formulation of the query. The proposed user interface includes

these features and thereby provides the user a means to incrementally formulate his query. The interface provides this capability in a flexible and efficient way.

The capability to easily iterate over query formulation steps must be integrated into the design of a DBMS user interface. As the user works through the process of formulating a complex query he must feel free to use trial and error. If he realizes a mistake, it should be easy to go back and try again without having to redo the work it took to get him there. The user approaching the database with a need to explore must be encouraged to do so. A simple and efficient means to repeat a step with a capability to introduce a small change does this. There are several features which should be included in a database user interface enable and facilitate iteration. These features are presented as features 11 through 13 below. These features all involve facilitating the action of iterating or repeating steps while formulating the query.(KUNTZ89, WONG82)

i. Feature 9 : Manipulation of Data Flow to Achieve Objective

As the user formulates his query he begins at one place, does certain things, and then ends up with a result. The correctness of and ease of achieving the result is in a large part dependent on the clarity with which the user can progress through this process. The proposed query formulation paradigm is based on a very simple concept of data flow. The

user begins each step of his query by specifying the relations which contain information which is relevant to his query. This is the beginning data. The user then applies various operations to this data. The result of these operations is always a relation. Data has in effect flowed from the one or two original sources to the resulting destination. This idea of data flow is very easy for the user to grasp.

j. Feature : 10 Use of Simple Operations

As mentioned in feature 9, during the query formulation data flows from a set of one or more relations to a resulting relation. During this flow, some sort of operation occurs on the data. This operation must be completely within the control of the user. The operation must also be simple and clear to the user. For this reason the basic relational algebra operations are used as a basis for these operations. These basic operations have been augmented to enhance and facilitate their use. These enhancements are demonstrated in the implementation description contained in chapter 4. When presented in a framework of data flow the actions of these basic operations becomes very simple and clear to the user. The user maintains his feeling of being in control and of working with an easily understood piece of his query.

k. Feature 11 : Piecemeal Design and Construction of Qawry

The DBMS user interface must permit piecemeal query formulation so as to minimize the scope the user must

mentally deal with at any single moment. This reflects the way people think when confronted with a complex or unclear problem. This notion of piecemeal construction is supported in several ways. The user interface allows the user to save portions of, or entire finished queries. These pieces can then be retrieved for use in the construction of the current query or subsequent queries. These query fragments can be linked or glued together to prevent the need to reinvent them. This concept of saving, manipulation and linking of query fragments is very similar to work done by other researchers (KUNTZ89, WONG82). What sets this concept apart here is the way this idea is integrated with data flow, simple operations and the graphical environment.

1. Feature 12 : Saving and Retrieval of Previously Defined and Commonly Used Joins

There are relationships the user observes when viewing the database schema. These relationships are included in the database schema because at the time of the database design they modeled a real world relationship. An example of this is the relationship "EMPLOYEE WORKS_FOR DEPARTMENT". In the database schema the user observes two relations and a relationship between them (i.e. WORKS_FOR). The user might use this relationship between EMPLOYEE and DEPARTMENT on a frequent basis. To prevent the user from having to keep conducting the join operation each time he wants to use this relationship, the saving and retrieval common joins is

possible (KEIM91). This is similar in function to saving query fragments as mentioned above. The difference is this separate classification allows the aggregation of a special type of predefined operation in order to facilitate the schema-to-query transition forawhe user. This approach falls somewhere between the explicit specification of joins as in SQL and the approach proposed by Elmasri and Larson (ELMASRI85). Elmasri and Larson propose the system automatically assume the desired relation. Their work is discussed in more detail in Chapter 2. The benefit with this approach is it eases the work involved in using a relationship, while at the same time not compromising the users sense of being in control.

m. Feature 13 : Immediate and Meaningful Feedback

As the user is formulating his query in a stepwise fashion he must be enabled to get immediate and meaningful feedback. After each intermediate step in the query formulation process the user must do one of two things. Either redo the current step or to move on to the next step. Meaningful feedback is necessary in order to assist the user in deciding which way to go (KUNTZ89). The proposed interface provides two examples of this feature. The first example is the capability to view results of a query at any point in the process. At any point the user may request to see results which causes the system to open a results window. In the window the user can see the data displayed in a default report format. This display can then either be removed from the

screen or retained. If retained, a representation of the state of the query at the point of execution is retained along with the result window. The availability of such feedback allows the user to quickly evaluate his progress and thereby decide if he should go on to the next step or go back and repeat the step. This capability also promotes exploration of the database. The user can place the results windows of two queries side by side and hence allowing him to see the results of the difference in the two queries.

Another example of intermediate feedback involves the display metadata. As previously mentioned the user canawoggle on and off the display of metadata. Also previously mentioned was the notion that the result of every operation is a relation. Combining these allows the user to ascertain useful information such as the numbers of rows and columns in his resulting relation at each juncture in his query formulation. This type of feedback serves as useful information to the user as he tries to evaluate his progress.

4. Minimize the Effort Required of the User

Effort required of the user refers to any work which the user must perform in order to accomplish his objective. This includes anything which causes the user to spend time, perform movement, or memorization. Many of the features already mentioned serve to alleviate the effort required by the user. Allowing selection by clicking on objects is a prime

example. There are several other features which should be a part of a good DBMS GUI. These are discussed below.

n. Feature 14 : Ensure the Earliest Detection of Errors

When performing a task of any significant complexity there are always a number of errors which can be made. A person performing such a task for the first time will make more errors than someone very familiar with the task. This is the nature of performing queries against a relational DBMS. A good DBMS must detect the errors as close to when they occur as possible. This will tend to minimize the lost time suffered by the user as a result of the error. It does not make sense on a lengthy multi-step query to notify the user after he is complete that he made a critical mistake in the first step. The sooner the user finds out the less the cost to him. In addition to detecting and notifying the user of the errors, the interface should also provide the user with the likely cause and potential solutions. These last two things are not always possible as the system cannot easily predict the users intentions. Where possible however, they should be provided.

There are a number of ways the proposed interface incorporates this feature. An example is seen when choosing relations upon which to perform an operation. A unary operation must have only one operand selected and a binary operation must have two operands. A violation by the user

results in an error message, a brief explanation, and an opportunity to correct the problem. A second example relates to the join operation. If a user chooses two join attributes which are of a different data type, then the system immediately lets the user know. In this case an error message explaining the error and a suggested fix is provided. These are the types of errors which the user would be better off knowing about right away. As well as alerting him while the issue is still hot in his mind, this feature protects against jeopardizing future steps in the query.

o. Feature 15 : Automatically Make Necessary Tools and Information Available

A user interface for a DBMS should make the users job as easy as possible. One way to do this is to know where the user is at and what he is trying to do. With this information the system can sometimes predict the tools and information the user needs. This is like an assistant working for a carpenter. If the assistant is paying attention, the carpenter will seldom need to ask the assistant to hand him a tool. Not having to ask and wait for a tool saves the carpenter time. With a DBMS user interface the system is analogous to the assistant. The user interface should provide the user with those things he may need, without the user explicitly having to request everything.

An example in the proposed user interface occurs when the user requests a select operation. The user first

chooses a relation on the QMF work surface. The user then clicks on "PICK ROWS" from the TOOL BOX menu. The user interface then automatically provides three pop up windows. One window is the text window where the row conditions are displayed as they are built. The second window contains a list of all the attributes from the chosen relation. The third window provides a list of all the operations the user might need to choose from (e.g., equals, greater than, less than, and, or). At this point the user does not need to ask for anything. All he needs to do is point and click, unless of course the user desires to type in a constant value. Even in the latter case the user interface can provide information as described in the next example.

An example of automatically providing the user with necessary information is present while formulating a query as in the example above. Presume the user is selecting rows and building the selection condition. If the user chooses the attribute DEPARTMENT and then chooses the operation EQUALS, the system immediately pops up a window containing the names of the twelve departments in the organization. The user is now able to avoid manually typing in the department name as well as the memorization and occasional misspelling which occur when humans must manually type values. This type of value window is not feasible in all cases due to the inherent uniqueness in some values. Imagine working with a relation of ten-thousand distinct employees and doing a selection on SSN.

A pop up window of ten-thousand values is not very helpful. In the proposed interface, such comparisons on attributes with twenty or fewer distinct values in the database is suggested. A parameter such as this however, should be able to be customized for the user. (KUNTZ89)

p. Feature 16 : Stream-line Repetitive Actions

There are a number of actions which are frequently performed by the user. Whenever possible a good user interface should provide a way to cut down on the need for the user to repeat his actions. This can be accomplished by providing shortcuts for the veteran user or by remembering the repetitive actions and allowing the user to re-execute them by a single action.

A trivial yet important example in the proposed interface involves the saving and retrieval of queries and query fragments. The nature of queries posed by most users are similar in structure. It saves the user a lot of time if all he needs to do is pull up yesterday's query and make a single change and then re-execute it. This idea is similar in function to that mentioned in ISIS (GOLDMAN85).

Once a user establishes a pool of frequently required queries, significant time can be saved by pulling up an old query and making a few changes. Whenever possible this sort of time saving capability should be provided.

q. Feature 17 : Use a Default Result Format

The display of data is a central and much used part of the query formulation process. Display of data may occur both during the query specification and after the query is complete. Because of this and because it is desirable to minimize the users effort, a user interface should make displaying data an efficient and simple process. Concerns about how the display of data should look, must not take place during query specification. When the user is working through the process of correctly formulating a query, he should not be burdened with concerns of how the data should be displayed. Once the query is at a desired state the user can pretty up the display if desired. The functions related specifically to how the data should be formatted should be separate from the functions of getting the correct data. This means there should be a default display format. There should also be a means to manipulate the format of the displayed data once the query is defined. These two capabilities support a separation of the query formulation process from the process of formatting the display of data. Some interesting research has been conducted in the area of displaying database objects, but this area is outside the focus of this thesis (MAIER87). This is not to say that display of results is not important to the user performing a query. The important point is that specialized display of data and query formulation should be conceptually separate activities from the users perspective.

An example of a simple and efficient means of displaying data is present in the proposed DBMS user interface. The user of the Query Management Facility (QMF) is frequently going to be looking at data through use of the Results window. Whenever the user requests to see the results of the query a results window is displayed with the data in a default format. The user then has two ways of changing the display format. One way is to sort the data. The proposed interface allows the user to sort the data either in ascending or descending order. The user can also indicate multiple sort fields. The second way the user can change the display format is by rearranging the sequence in which the attributes are displayed. The user can for example, cause the third column of data to be displayed in the first column, or the last column, or in any other position. These two methods of altering the display format are in addition to being able to move and re-size the Result window. These last two are functions of the window manager and are not considered a part of the user interface design.

C. APPLICABILITY TO MULTIMEDIA DATABASE SYSTEMS

The approach taken in the multimedia database project implies that multimedia data can be handled in the same manner as formatted data (LUM89). From the users perspective, a given multimedia object (e.g., photograph) is functionally no different from textual data, such as the text representing a

persons street address. The user can store, query, and display either type data. As far as the database system is concerned, the photograph is of an abstract data type (e.g., photo) whereas the address is of a formatted data type (e.g., character). The system may require specialized ways of displaying and manipulating multimedia data. From the users view, this fact, does not pose any special considerations aside from those related to data display. Further discussion of these concepts can be found in related works (LUM89, KEIM91, MEYER-WEGNER89, KIM91).

Given the concepts mentioned above, it is easy to see that a multimedia database is a special case (i.e., subset) of a traditional database. This implies that all of the criteria and features presented in this chapter are directly applicable to the user interface for a multimedia database system. The unique user interface characteristics required of a multimedia database system consist of those related to data display. The means of displaying and working with multimedia data should be designed as consistently and functionally similar as possible to the means used for display of formatted data. By careful application of the features presented in this chapter, the designer of a multimedia database user interface can be assured that he is designing a good user interface.

IV. DESCRIPTION OF GRAPHICAL QUERY MANAGEMENT FACILITY

The proposed user interface is described by means of figures contained in this chapter. These figures represent what one might see on a computer screen while using the user interface to perform database queries. First a general description of the major functional parts of the interface are described. This is followed by a description of a simple query. This is intended to illustrate the basic functioning of the Query Management Facility (QMF). After the simple query the issue of complex queries is discussed along with examples. This chapter is completed with a discussion of the aggregate functions. The sample database schema as well as some of the queries used as examples, were taken from a textbook written by Elmasri and Navathe (ELMASRI89). Modifications were made to the schema and queries to fit the purposes of this thesis.

A. MAJOR FUNCTIONAL PARTS OF INTERFACE

To perform a database query the user enters the portion of the interface called the QMF. When using the QMF the main retrieval window is always displayed. This main retrieval window is presented in Figure 1.1. When a particular database is opened, the name of the database is displayed on the top title bar. In Figure 1.1 this name is "COMPANY DATABASE". The "OPEN DATABASE" button on the lower menu bar is used to either

open a database or to open a different database in the case one is already open.

The "TOOL BOX" area is the place where the user goes to select any operations he is to perform. These operations fall into two logical groups. These groups are "table" operations and "get" operations. As you can see in Figure 1.1, the Tool Box is arranged accordingly.

The TABLE OPERATIONS are performed on tables or in relational terminology, relations. As discussed in Chapter 3, all operations have one or more tables as input and a table as output. The user has a scroll bar to get to operations not shown. Those operations used most frequently are displayed first. These are the relational selection, projection and join. The names in the tool box are put in more simple terms for the non-technical person (i.e., Pick Rows, Pick Columns). These operations are followed by the basic set operations union, intersection and difference. These work as one would expect. The set operations are followed by a grouping operator, a containment operator and then some aggregate functions. A more thorough explanation of the behavior of theaw operations can be achieved in the example queries which are to follow.

The GET OPERATIONS area contains those operations which enable the user to go out and get a resource for use within the query. These resources include previously defined joins, previously saved queries, and relational tables of data. From

this point on, these tables of data (i.e., relations) will be referred to as tables.

The SHOW QUERY window is the area where the user sees objects representing his query grow from the original table or tables to the final result of his query, which is also a table. Scroll bars are provided to facilitate control of the area which is actually viewed in this window.

The USER ACTION bar is an area the system uses to suggest appropriate actions to the user. The user will not look to this area when using the HELP option or for information when errors are detected by the system. Separate pop-up windows are used as help windows and error handling windows.

The use of the mentioned features, as well as the remainder of the features of this main window will be demonstrated by their use in the query examples to come.

B. SIMPLE QUERY WITH INDIRECT USE OF MULTIMEDIA DATA

The simple query is a good means to illustrate the screens a user would see while performing a database query. The query which will be performed is: "Retrieve the photograph and address of the employee whose name is John Smith.

The user begins with the screen depicted in Figure 1.1. Since in this example a database is already open, a pictorial view of the database is displayed as illustrated in Figure 1.2. Whenever a database is open, the user automatically gets

the "SHOW PICTURE OF DATABASE" window (see Figure 1.2). Details of this window will be discussed at a later point.

By considering the query, the user realizes that EMPLOYEE is the table in the database (Figure 1.2) likely to contain the information he is looking for. If there is doubt he could explore the schema more to further convince himself or he could just go ahead and choose the EMPLOYEE table and change it later, if it proves to be wrong. Although this example is trivial as far as queries go, the point is the user interface is designed to expedite the work of the familiar user by making everything easily available. It is also designed to encourage the unfamiliar user to explore the database and use trial and error without a big loss in efficiency.

The user places the cursor on the object labelled EMPLOYEE (Figure 1.2). This causes the table to be selected and placed into the SHOW QUERY area of the main window. The state now is that depicted in Figure 1.3.

The user now has the table which has rows of information, with each row representing an employee. The user knows that he is only interested in a subset of these rows, so he selects the "Pick Rows" operation from the Tool Box. This action causes an "operations" pop-up window and "Pick Rows Condition" pop-up window to appear. The Operations window contains the comparison operators and the logical connectives the user may need in specifying the selection condition. The Pick Row Conditions window displays the actual selection condition as

specified by the user. These windows are illustrated in Figure 1.4. These windows can be moved around and so forth in accordance with the style and techniques of the window manager the user happens to be using. The functioning of the window manager is independent from the DBMS user interface (HELLER90a).

The user now has the job of communicating which rows he is interested in (i.e., employee named John Smith). One option the user has to go to the get data table and get a tabular menu of the data (i.e., attribute names) of the EMPLOYEE table. If this were an intermediate table he was working with, this tabular window would automatically appear in a pop-up window. At this point in the example query, there is a better way to choose the required attribute name. As stated, the picture of the database schema is visible to the user. The user goes to that window and selects the SHOW FIELDS button. This causes the fields (i.e., attributes) for each entity to be shown. The cardinality of the relationships could be and in this example are selected to display the cardinality of the relationships in the database schema. All of these features are displayed in Figure 1.5. You can see that the selected buttons are highlighted in figure 1.5 corresponding to the users choice of what he wants to see displayed.

The user selects the attribute labelled FNAME which corresponds to the users first name. This is shown in Figure 1.5 As the user makes this selection the attribute name is

placed into the Pick Row Conditions window (Figure 1.6). The user continues to choose desired attribute names, comparison operators, and logical connectives to build his selection condition. The only items that must be physically typed by the user are the attribute values to be used for comparison. In cases where less then a certain number of distinct values exist for an attribute, say fifteen, a pop-up window is displayed to permit mouse selection of the desired value vice typing. The certain number of distinct values is a user or database administrator set parameter. This process of communicating the selection condition progresses as indicated in Figure 1.6. When the user is finished he clicks the cursor back in the open area of the SHOW QUERY window. This brings him the screen shown in Figure 1.7.

In Figure 1.7 the result of the Pick Rows operation is indicated by a box placed below the EMPLOYEE box. The icon for the Pick Rows operation is also displayed. If the user wants to ever go back to modify this operation, he only needs to click on the icon or the corresponding result box. At this point two boxes appear in the Show Query window. Each box in the window represents a table which can be awed for further definition of the query. It should be noted that at this point the user could choose to look at the intermediate results represented by the newly created results box. He would do this by clicking on the box and then clicking the DISPLAY RESULTS button on the bottom menu bar.

The user has just the information about John Smith in the intermediate results box. He views this information as positioned in columns. The user does not want all the information about John Smith, just his address and photo. To trim the current result down to just the desired information the user first clicks on the result box to choose it as an operand and then chooses the PICK COLUMNS operation from the Tool Box. These action are depicted in Figure 1.8. As he chooses the operation a pop-up window appears with the attribute names for the selected operand. The system is smart enough to know the operand selected does not have a corresponding object in the database picture, hence the information in this window will be needed by the user to specify the parameters of the chosen operation. This pop-up window is shown in Figure 1.8

In Figure 1.9 the screen is shown after the user has chosen the attribute names corresponding to the fields he is interested in. It might be noted that the user has decided to keep the users name in the output from the operation. When the user is finished he clicks in the open area of the Show Query window which results in the screen shown in Figure 1.10.

At this point the user has the results he wants from his query. he clicks on the final result box and then selects the DISPLAY RESULTS button on the bottom menu bar (Figure 1.11). This causes a QUERY RESULTS window to pop up. The Query Results window is shown in Figure 1.12.

Multimedia objects are shown in a consistent manner whenever they are chosen for display. In Figure 1.12 you can see that for the employees photo (i.e., EPHOTO), a button is displayed in lieu of a value. Were this a voice recording or a piece of video, the same thing would occur. To see, hear, or cause the display of the multimedia object the user clicks the button as indicated in Figure 1.12. This causes the appropriate display, based on the definition of the abstract data type corresponding to the multimedia object. In this case the display is illustrated in Figure 1.13.

From the example provided by this simple query it can be seen how simple the process of formulating a query is. The user need focus on only one aspect of his query at a time. In this example the user concentrated first on choosing an appropriate table, then picking the desired rows, then picking the desired columns and finally on displaying the data. The user could have reversed the order of the row and column operations. The point is the operations are simple, flexible and responsive to the way the user thinks. Because of all the graphical tools given to the user, the low-level of the operations was not a labor intensive factor.

The marriage of the low-level operations and the graphical capabilities allows the important benefits of low-level operations without the drawbacks. Chapter 3 discusses this point in far more detail.

C. SIMPLE QUERY WITH DIRECT USE OF MULTIMEDIA DATA

This example query demonstrates the direct use of multimedia objects in the query specification process. The query to perform is : "Retrieve the name and photograph of employees wearing U.S. military uniforms". The user begins with a screen such as that in Figure 2.1. The user realizes he wants information about employees so he chooses the EMPLOYEE object from the PICTURE OF DATABASE window as in the previous example. The result from this is shown in Figure 2.2. In Figure 2.2 the user has information about all the employees in the company. The user knows he wants information on only the employees wearing U.S. military uniforms in their photograph, thus he chooses the PICK ROWS operation from the Tool Box. This choice results in the screen as seen in Figure 2.3. The user must next choose a field in the employee table on which to operate.

Figure 2.4 illustrates the user going to the PICTURE OF DATABASE window to select the field on which he wants to base his row selection condition. In this case the user is shown selecting EPHOTO (i.e., employee photo). This results in a pop-up window entitled "MultiMedia OBJECT SELECTION". This window is used to enable the user to enter a description of the multimedia object he wants to select. The topic of selecting multimedia objects is explained in a paper by Lum (LUM89) and is further discussed in other papers by Kim (KIM91) and Keim (KEIM91). How and why this technique is used

is not further discussed in this thesis. The interested reader is directed to the references.

The user must enter a description to select the multimedia objects corresponding to the photographs of employees wearing U.S. military uniforms. The user decides to phrase the description, "Person wearing a uniform". The user then clicks the TEST button on the menu bar. This causes the appropriate search and a pop-up window entitled, "MultiMedia OBJECT SELECTION" results. This is all illustrated in Figure 2.5. This window resulting from the search is designed to give the user feedback on the progress of his multimedia object selection. In Figure 2.5 you can see the window displays the number of objects selected, in this case four. The window also has one entry for each object selected in the search. Each entry displays the original description which was stored with the object. This can be used to assist the user in tailoring his object description. Each entry also has a SELECT button. These are all set to on (i.e., highlighted) when this window is first displayed. The user has the choice of scrolling through these entries and indicating which multimedia objects he wants selected by toggling these SELECT buttons on or off. The user can also choose the, "Try a Different Description" button to iterate over the process of entering a description. In this example say the user notes in Figure 2.5 that too many objects were selected. The user also notes one of the descriptions which mentions a guy wearing a Turkish Navy

uniform. This clues the user in on the fact that he was too general and vague on his description of what objects he wanted. The user selects the Try Different Description button and is returned to the window for entering a description. The user enters a different and more precise description. He then tests the description with the result displayed as the two left windows illustrated in Figure 2.6. In Figure 2.6, the SHOW buttons are shown in the "on" state. This feature is provided to allow the user a quick means of verifying the correctness of his selected multimedia objects. The user has all the windows shown in Figure 2.6 on his terminal screen and is comfortable with his multimedia object selections. The user chooses the QUIT button in the main description window which automatically closes the child windows.

The user is returned back to the PICK ROW CONDITIONS window as depicted in Figure 2.7. This is an example of a user-to-system dialogue which is brought to a closure. The user is now completed with the selection of the multimedia objects associated with EPHOTO. Figure 2.7 indicates to the user that in this case 15 EPHOTO objects were chosen.

The next step in this query involves the user selecting the columns he wants to include in the results. This is shown in Figure 2.8. The user then requests the display of the results as shown in Figure 2.9.

Figure 2.10 shows the resulting QUERY RESULTS window. The user now knows that his query is complete and decides that he

wants to adjust the display of his output. The important point here is that the user did not have to consider the output display during the query specification. This is kept as a separate task.

In the example shown in Figure 2.10, the user desires to have EPHOTO be displayed as the last column in the output window. This is accomplished by selecting the column to move by clicking on the attribute. The user then positions the cursor where he wants the column to be placed and clicks the mouse and the column gets moved. The user also wants the output to be sorted differently. He wants to have the output sorted by last name. To do this the user clicks the SORT button on the lower menu bar which causes the SORT FIELDS pop-up window as seen in Figure 2.10. The user then selects the desired sort field or fields. Figure 2.11 illustrates the output resulting from these changes. It also shows the display of one of the multimedia objects.

D. COMPLEX QUERY

It is often difficult for a user who has a complex query in mind, to precisely and accurately communicate that query to the system. As discussed in Chapter 3, low-level simple operations, along with the simple concept of data flow, and the capabilities of a graphical environment make this process easier.

A complex query will serve to demonstrate this idea. The query which will be used is : "Find the names of employees who work on all the projects controlled by department number five".

Specifying this query in a linear programming language such as SQL is difficult. Even with an experienced SQL user it is difficult to specify the complex query. The user is forced to fit his query into his understanding of what will and will not work in SQL. Two possible SQL statements to specify this query are shown in figure 3.1.

Figure 3.2 shows the final screen the user would see using the QMF. When considering the query two things jump out at the user. One his he needs all projects controlled by department 5. The two upper right boxes in Figure 3.2 show the results of attaining projects controlled by department 5.

The second thing that the user sees he will need is a list of the projects which each employee works on. He can see in the picture of the database, exactly what fields are in what tables. The user notes that the table Works_On contains both the SSN of each employee working on a project plus the related project number. The user simply chooses this table and groups it on ESSN. This is a logical grouping which still results in a relation. This special quality of being logically grouped is pictorially represented with a meaningful picture of boxes grouped within the result box.(Figure 3.2)

The user knows the box on the right contains all the projects controlled by department 5. The user can now see that he wants to select the groups of employees on the left which contain all of the records in the box on the right. This means that he chooses the CONTAINMENT operation from the Tool Box.

The containment operator interacts with the user in an interactive style. It identifies exactly what type of containment the user is interested in. This means the user could choose only the groups containing exactly the same tuples as those in the second operand. He could also choose only the groups which contain at a minimum, all the tuples as those in the second operand. The user could also define the precise degree of overlap in terms of a discrete number of tuples in the second operand which must match those in the group in order for the group to be selected.

An additional feature which adds power to this containment operation is the user is able to designate exact attributes which are to be used in the containment operations definition of "match". In this query example for instance, the user has chosen to use only a single attribute from each group (i.e., PNO to match against PNUMBER). The dialogue environment allows for an efficient specification by the user of his desired intent.

To complete this query the user only needs to join the result with EMPLOYEE to pick up the employees name.

Another feature which will benefit the user working with complex queries is an ability to implode selected areas of the picture of his query. This is a way to simplify portions which the user no longer desires to see the detail. Once imploded, the symbol can be thought of as a black box as far as the user is concerned. If these are needs to be considered in detail the user is free to explode the symbol, thus retrieving the previous level of detail. Figure 3.3 shows the pop-up window resulting from the users choice to CHANGE DETAIL. Figure 3.4 illustrates the user circling the area of concern after which the user clicks on DECREASE DETAIL. Figure 3.5 illustrates the screen resulting from this action. As you can see, a level of detail has become hidden. The fact that the box represents a complex set of operations is still evident by the double framed box. This same tool can be used to build boxes representing complex query fragments of complex query fragments. The user is always able to go back and work with the necessary level of detail.

A second example of a complex query is shown in Figure 4. The query is : "List the project names for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project. Once again the SQL is shown in Figure 4.1.

The feature to be shown with this example is the use of the previously defined joins. When considering the query there are two pieces of data (i.e., tables) that the user knows

immediately he needs. One is the employees who work on projects. An unfamiliar user might go ahead and perform the join necessary to create this table. A user who is familiar with the database would know its a common join and thus he would use the GET PREVIOUS JOIN operation to get a menu of previously defined joins. He would do the same for Departments which Control Projects. Figure 4.2 illustrates how this can really expedite the definition of a complex query.

The use of previously defined queries works the same way. The user can decide to save whatever queries he chooses to. These Previous Queries can quickly be brought up and then either modified and ran, or executed as they are.

E. AGGREGATE FUNCTIONS

The aggregate functions include such things as count, sum, average, minimum and maximum. The provision of the aggregate functions within the user interface is a real convenience to the user. These functions do not impact the relational completeness of the query expression methods of the interface. They are merely a convenience. Such a convenience is a necessary part of a good query interface.

The aggregate functions can be applied to any table containing appropriate values upon which they can operate. The sum function for example, must have a numeric field upon which to perform the addition operation. The more interesting case of applying the aggregate functions occurs when they are

applied to a table which has been logically grouped. This case will be used in the following example.

The example query for demonstrating use of the aggregate functions is : "For each project, find the average salary, the average salary of males and females, and the number of males and females". The user begins the query with a screen as depicted in Figure 5.1.

The user begins the query by getting a table to work with. The user goes to the Get Operations area of the Tool Box and chooses the GET PREVIOUS JOIN operation. From the resulting pop-up menu the user chooses the EMPLOYEE-WORKS_ON-PROJECT join as the initial data table. By looking at the picture of the database the meaning of and fields contained in the table representing this join, are made clear to the user. Note that with this one simple step the user now has all the information he needs upon which to build his query. At this point the user has the screen as shown in Figure 5.2.

The user has a table with a row for each case of an employee working on a project. The user next applies the GROUP BY operation on this table, grouping the rows by first PROJECT and secondly SEX. This results in a SEX grouping within each PROJECT group. The screen resulting from this grouping is shown in Figure 5.3.

The users next action is to determine the average salary for each project and the average salary for each sex within each project. This is accomplished by choosing the AVERAGE

operation from the Tool Box. At this point the fact that the user is applying the operation to a grouped table causes a two step vice a one step dialogue. The first step which occurs is user identification of the field or fields upon which to perform. This step occurs with any application of the function, whether with a grouped or ungrouped table. The user specifies the target fields with a selection of the fields from a simple pop-up menu. Only the fields which are valid for this operation are displayed as highlighted in the pop-up menu. This ensures the user selects only valid data types as operands to the given aggregate function. For each field chosen the user goes through a second step.

The second step the user goes through is necessary since in this example the input table is logically grouped. The user must indicate whether the operation is to be applied to the entire table as a single logical entity or to some sub-entity. The grouping of the table allows a combination of possibilities. In this example the user can choose to apply the operation to the table as a group, to groups of projects, or to groups of sex within each project. The system is smart enough to determine the valid possibilities thus facilitating the users choice by offering only the valid options in a simple pop-up menu. In this example the user applies this operation to both to each project and to each sex group within each project. The user then finishes applying this operation

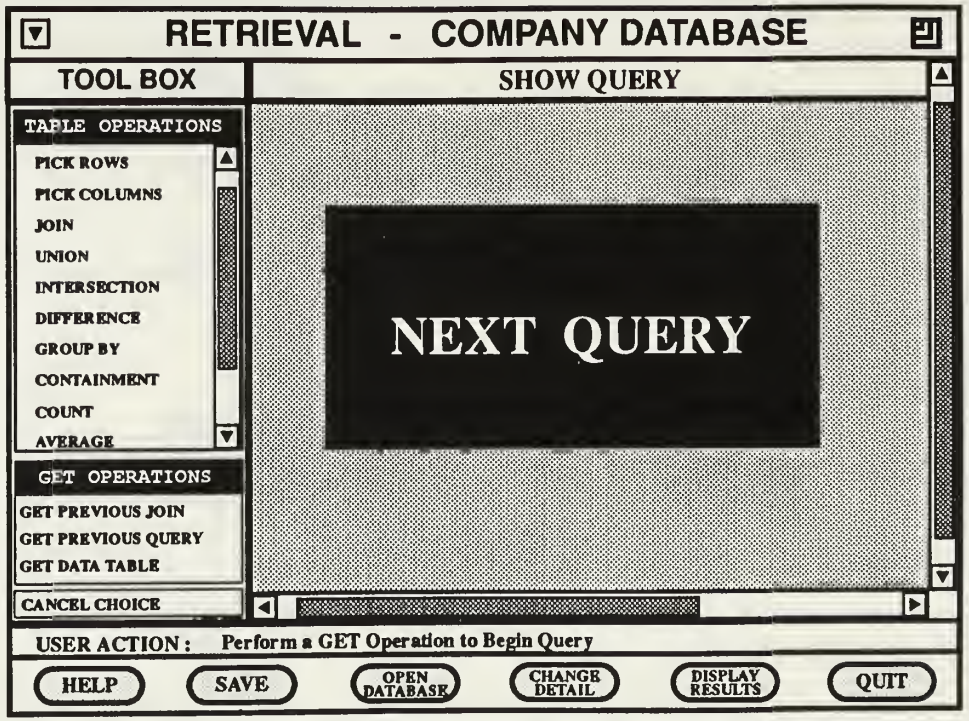
which results in a screen such as that illustrated in Figure 5.4.

In a similar fashion, the user applies the COUNT operation to the resulting table. The result of this action is seen in Figure 5.5.

The users only remaining action is to choose the attributes he wants in the result table. This picking of columns (i.e., relational projection) results in the screen shown in Figure 5.6. The field names chosen for inclusion are shown in the figure to demonstrate the types of default names the system assigns to the attributes created by aggregate functions. AVG_SALARY_PROJ contains the average salaries for each project while AVG_SALARY_PROJ_SEX contains the average salary for each sex within each project. In Figure 5.6 it can be seen that the user is finished building the query and has only to display the results.

Each of these aggregate functions is performed in an interactive manner between the system and the user. There are generally a number of possible ways to apply any given aggregate function. Only valid options are presented to the user thus minimizing guessing and possible confusion. The dialogue permits the user to back out from where he is at any point, or to proceed in the correct and desired direction. Each user action is facilitated by a simple menu allowing the user to pick from options with the use of a mouse. The user can feel comfortable that he is choosing a valid option. By

walking the user through the process of applying these aggregate functions, the system is more certain to ensure the user gets the results which he desires.



QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.1 Simple Query With Indirect Use of Multimedia Data

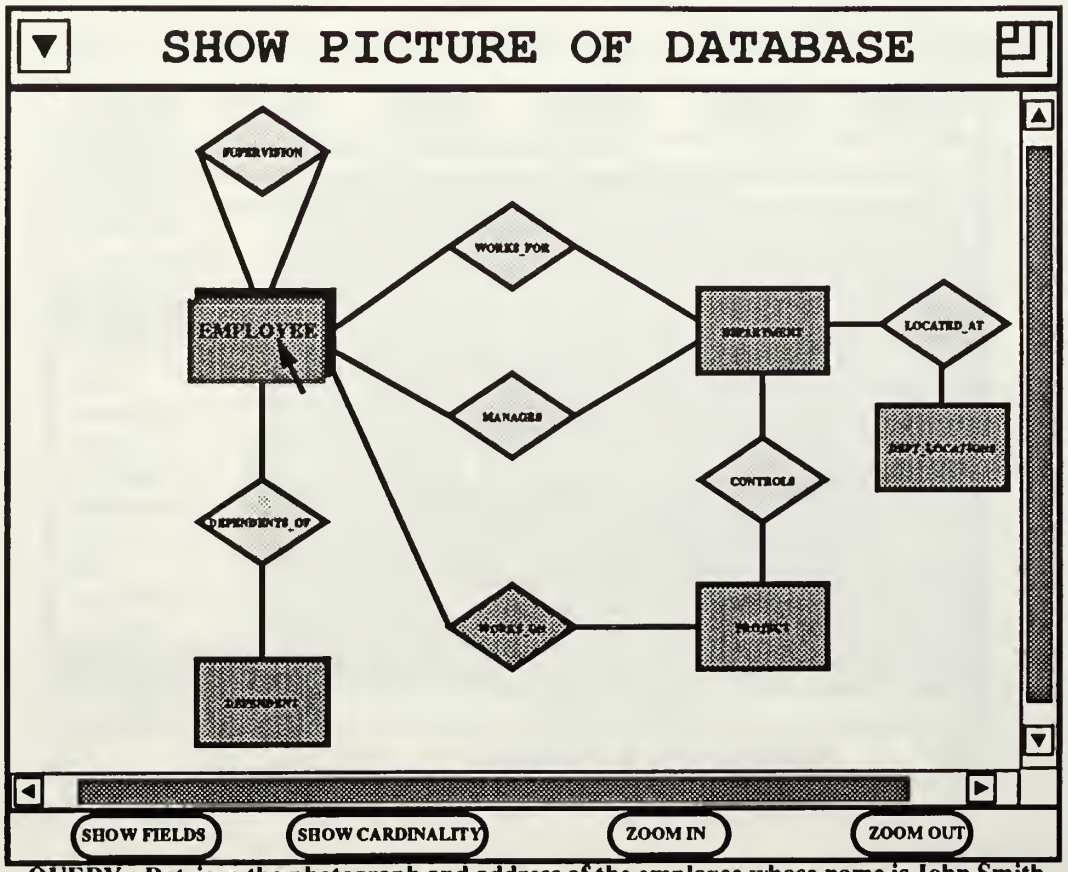
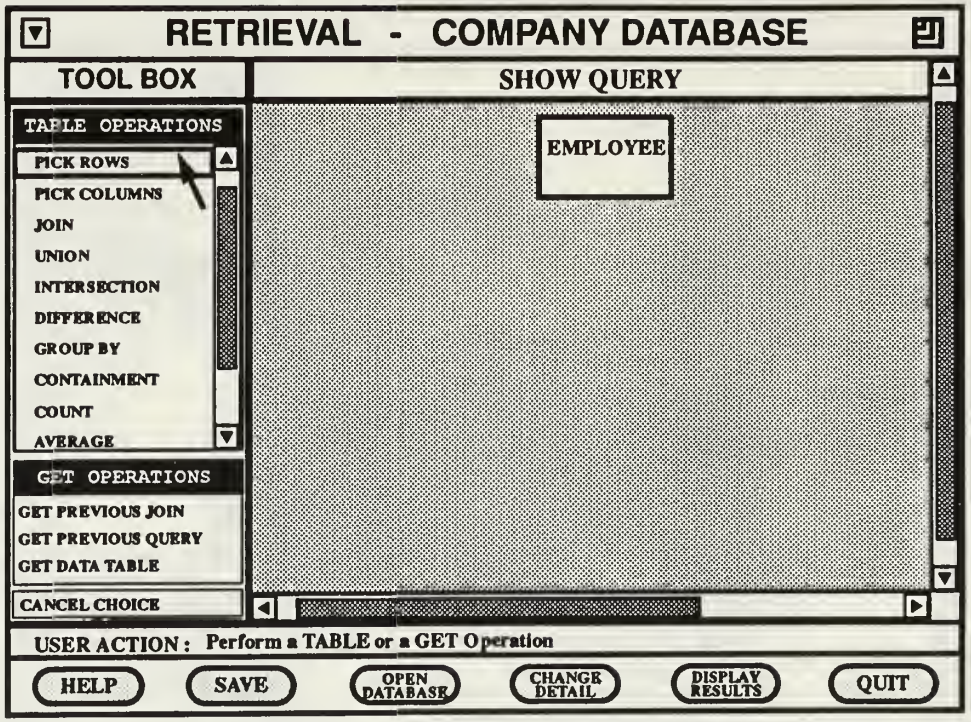


Figure 1.2 Simple Query With Indirect Use of Multimedia Data



QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.3 Simple Query With Indirect Use of Multimedia Data

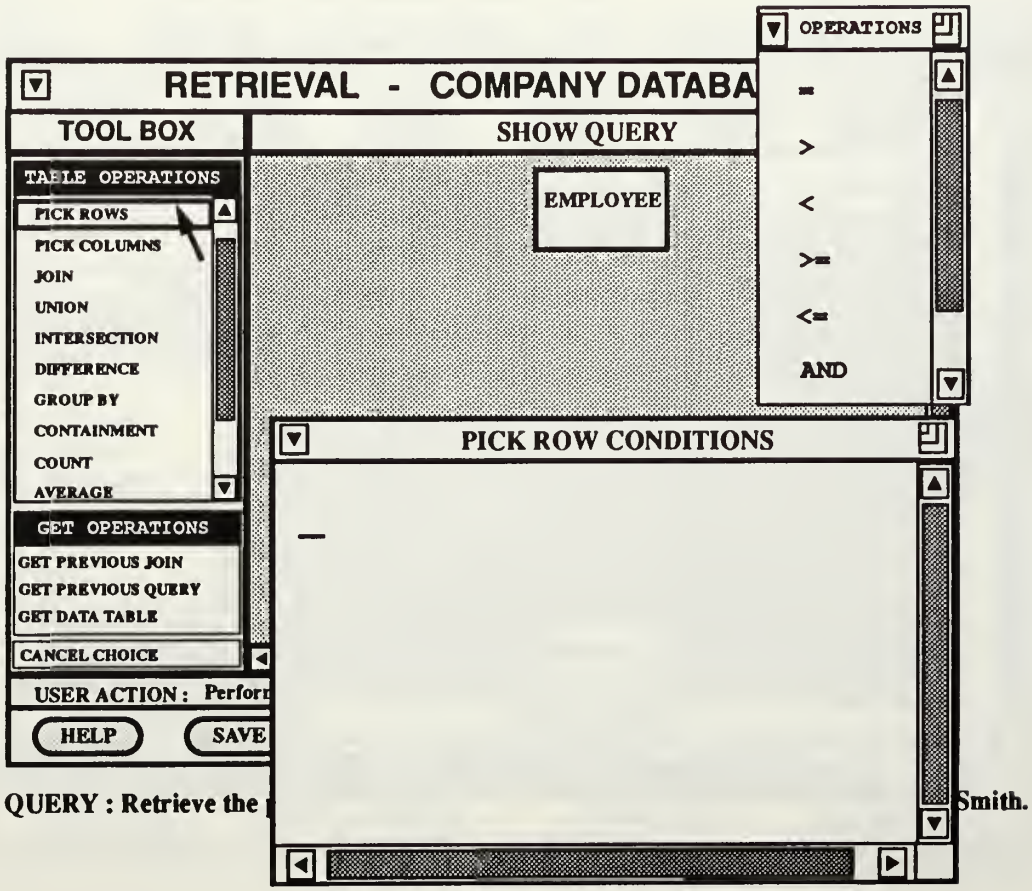


Figure 1.4 Simple Query With Indirect Use of Multimedia Data

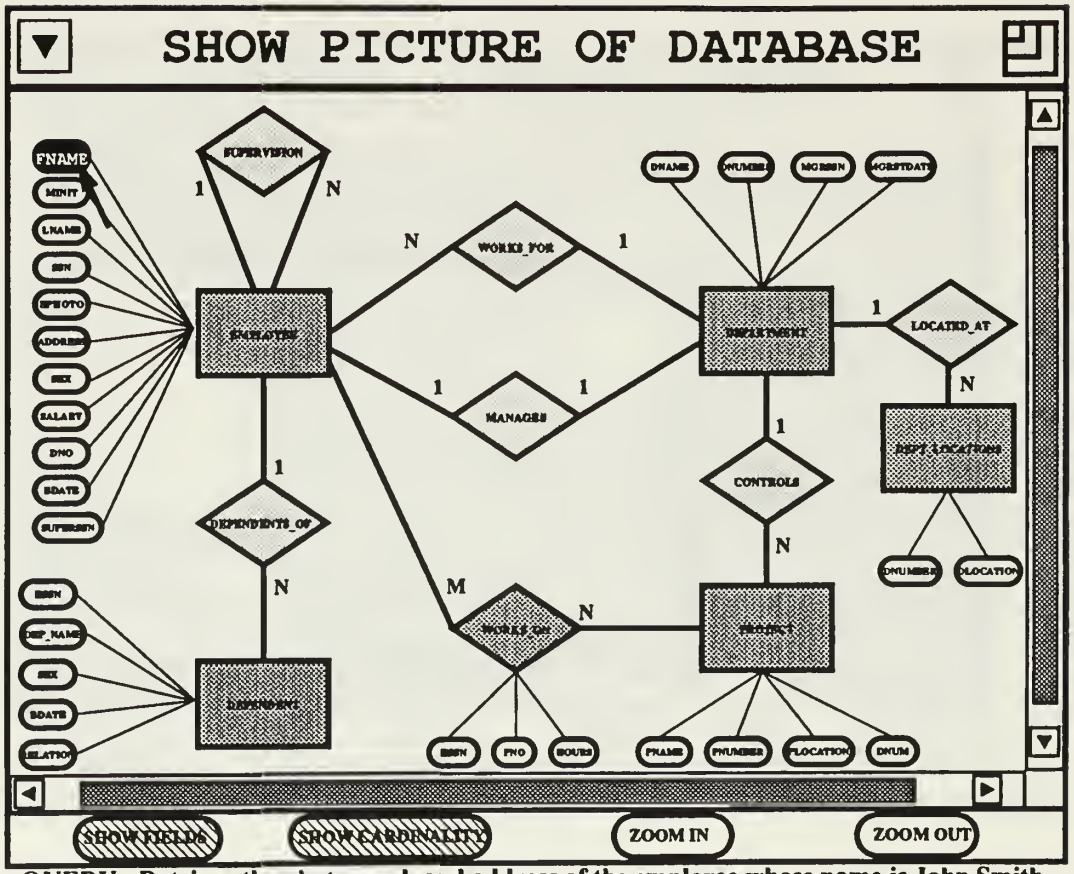
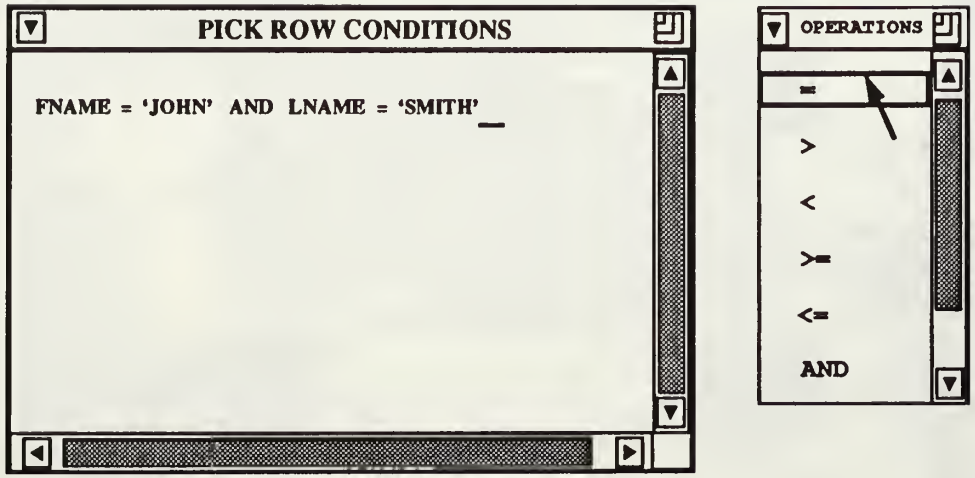
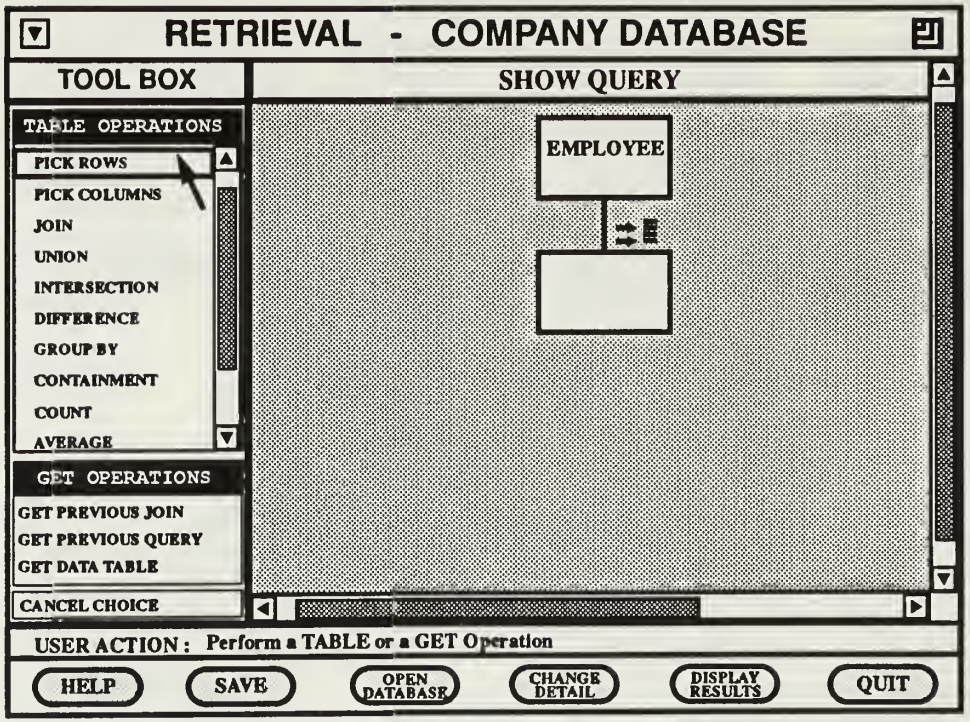


Figure 1.5 Simple Query With Indirect Use of Multimedia Data



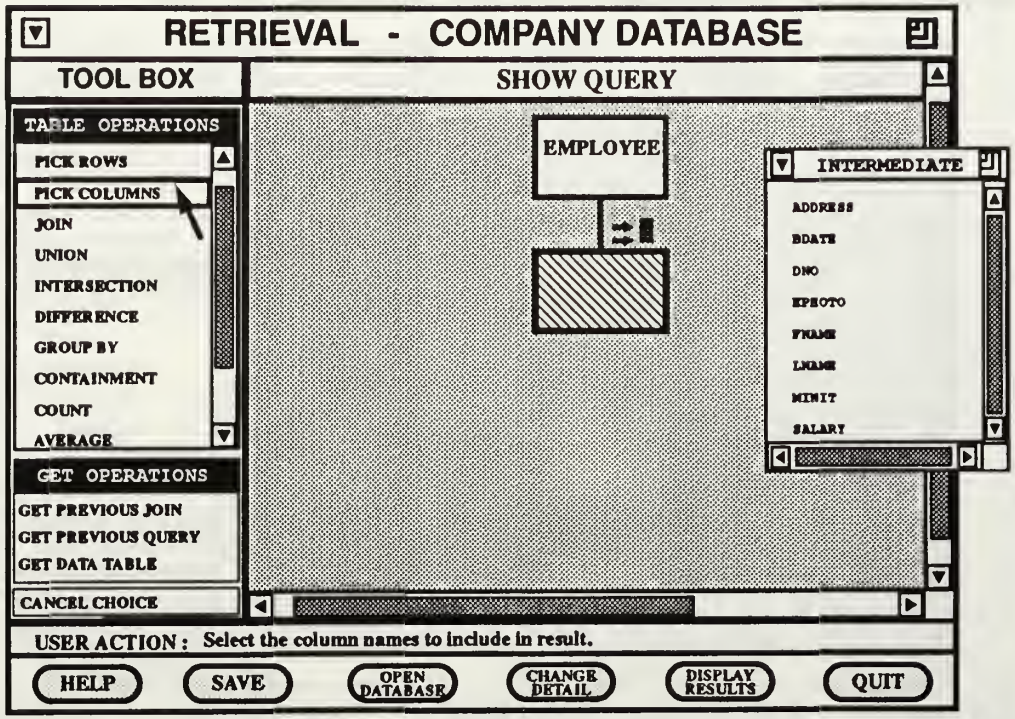
QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.6 Simple Query With Indirect Use of Multimedia Data



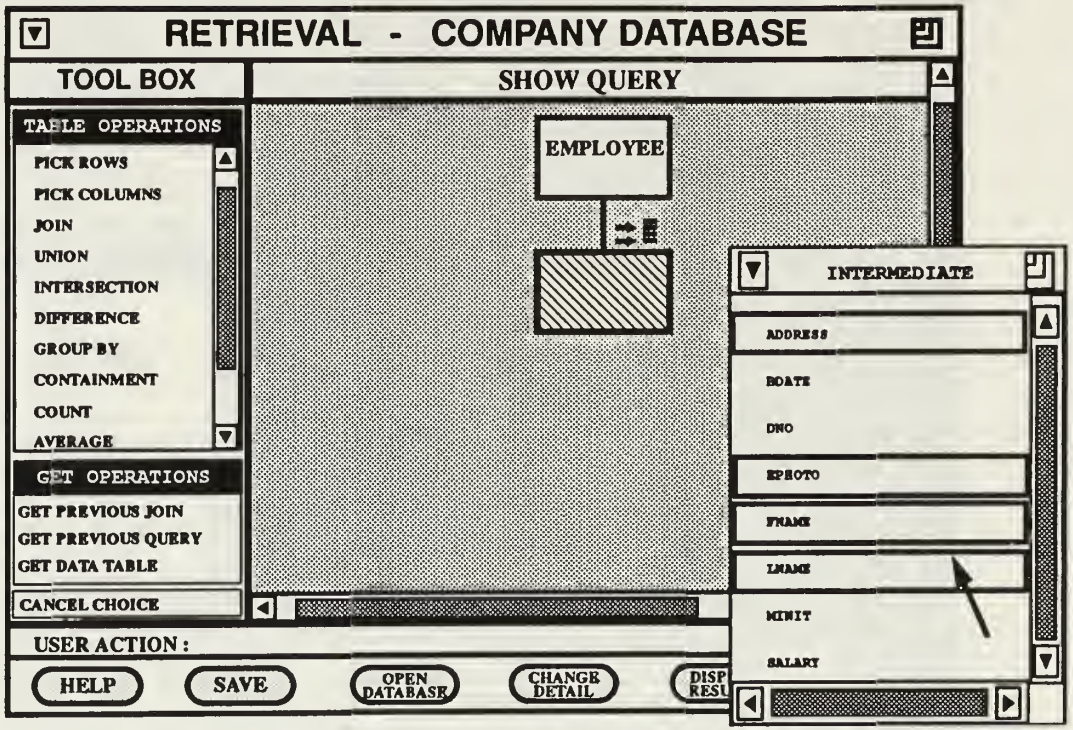
QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.7 Simple Query With Indirect Use of Multimedia Data



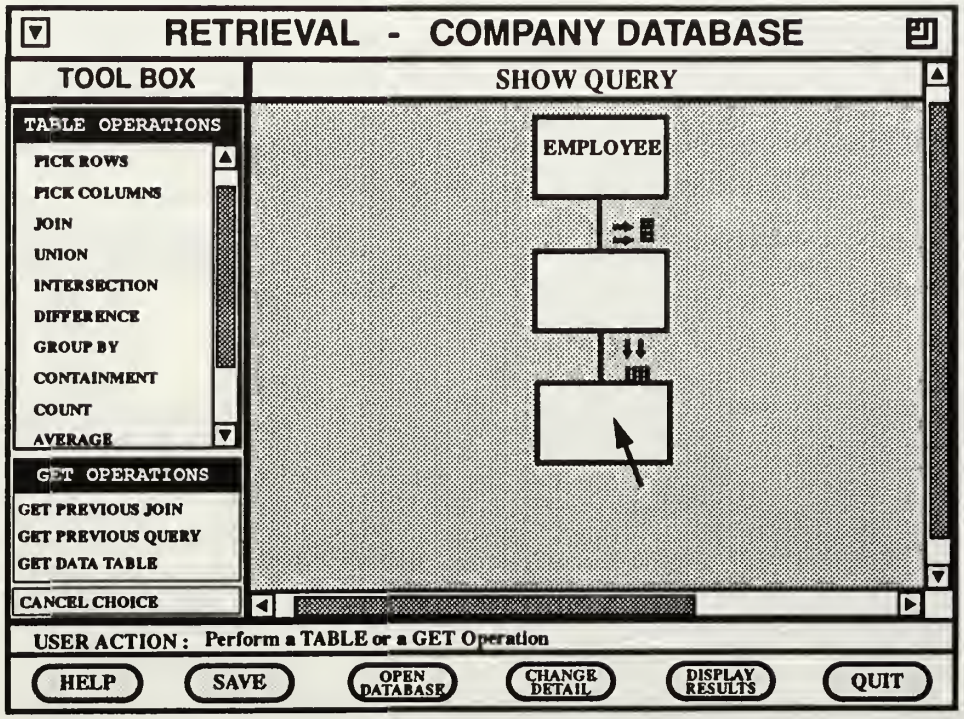
QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.8 Simple Query With Indirect Use of Multimedia Data



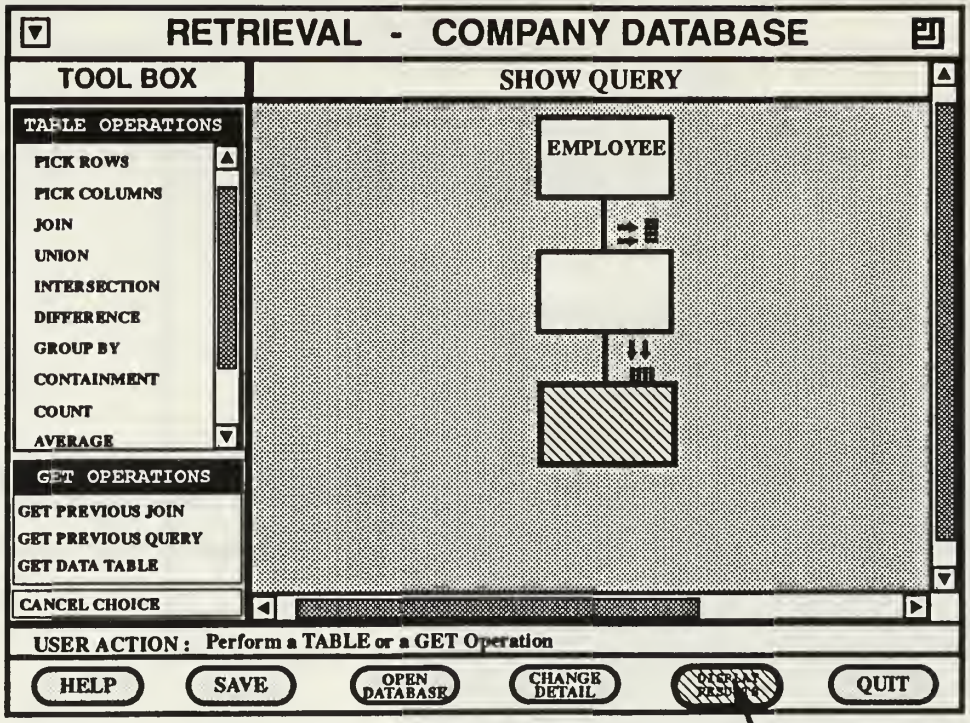
QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.9 Simple Query With Indirect Use of Multimedia Data



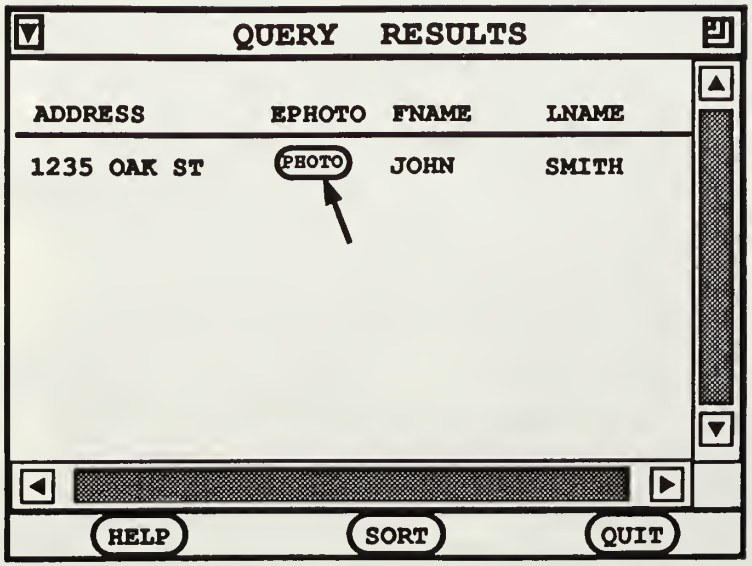
QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.10 Simple Query With Indirect Use of Multimedia Data



QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.11 Simple Query With Indirect Use of Multimedia Data



QUERY : Retrieve the photograph and address of the employee whose name is John Smith.

Figure 1.12 Simple Query With Indirect Use of Multimedia Data

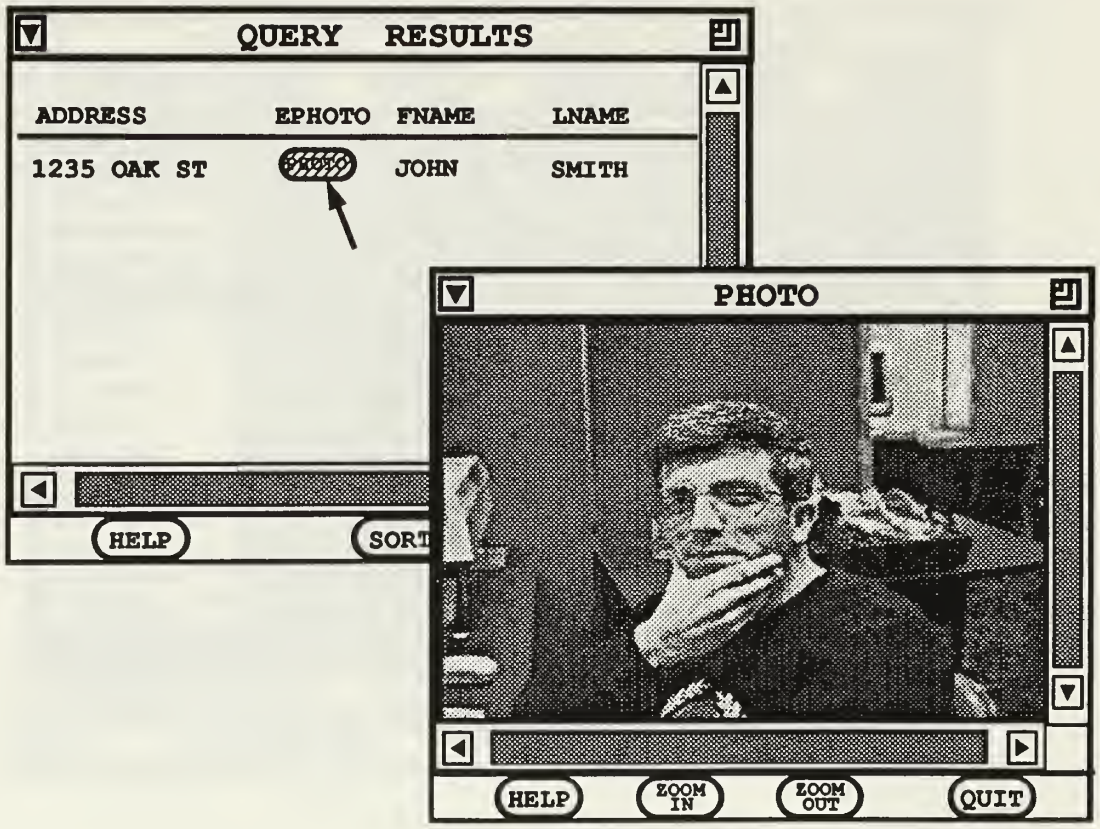
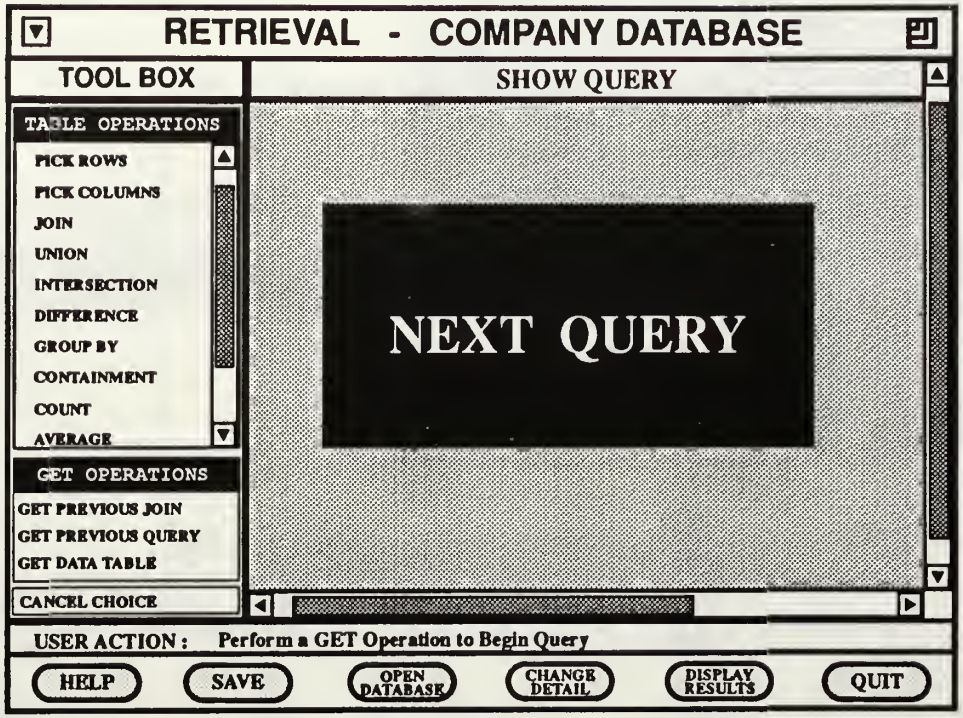
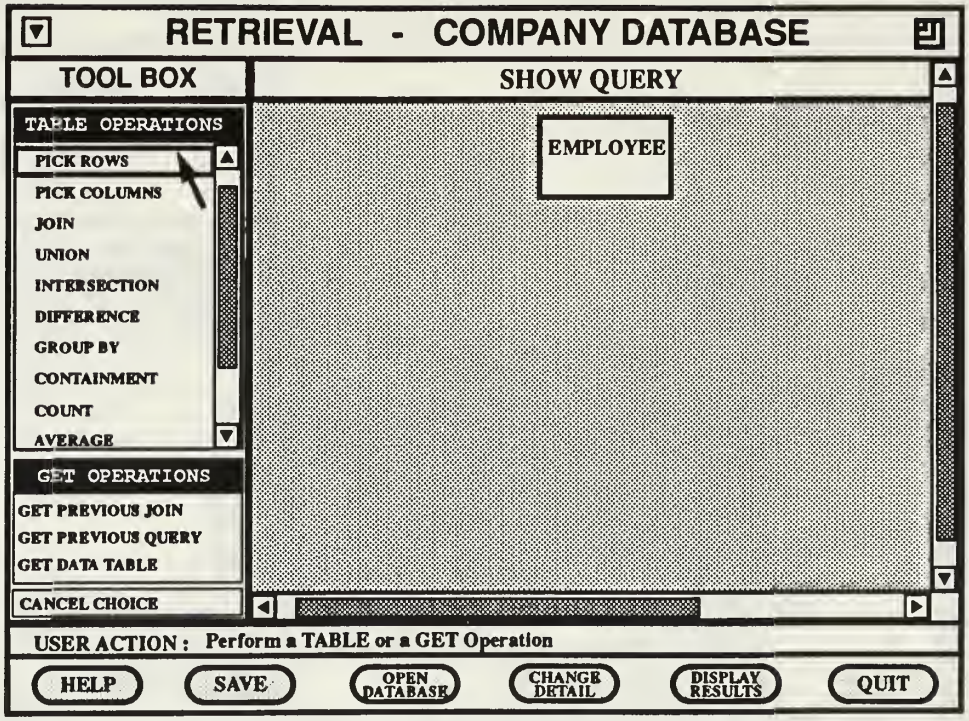


Figure 1.13 Simple Query With Indirect Use of Multimedia Data



QUERY : Retrieve the name and photograph of employees wearing U.S. military uniforms.

Figure 2.1 Simple Query With Direct Use of Multimedia Data



QUERY : Retrieve the name and photograph of employees wearing U.S. military uniforms.

Figure 2.2 Simple Query With Direct Use of Multimedia Data

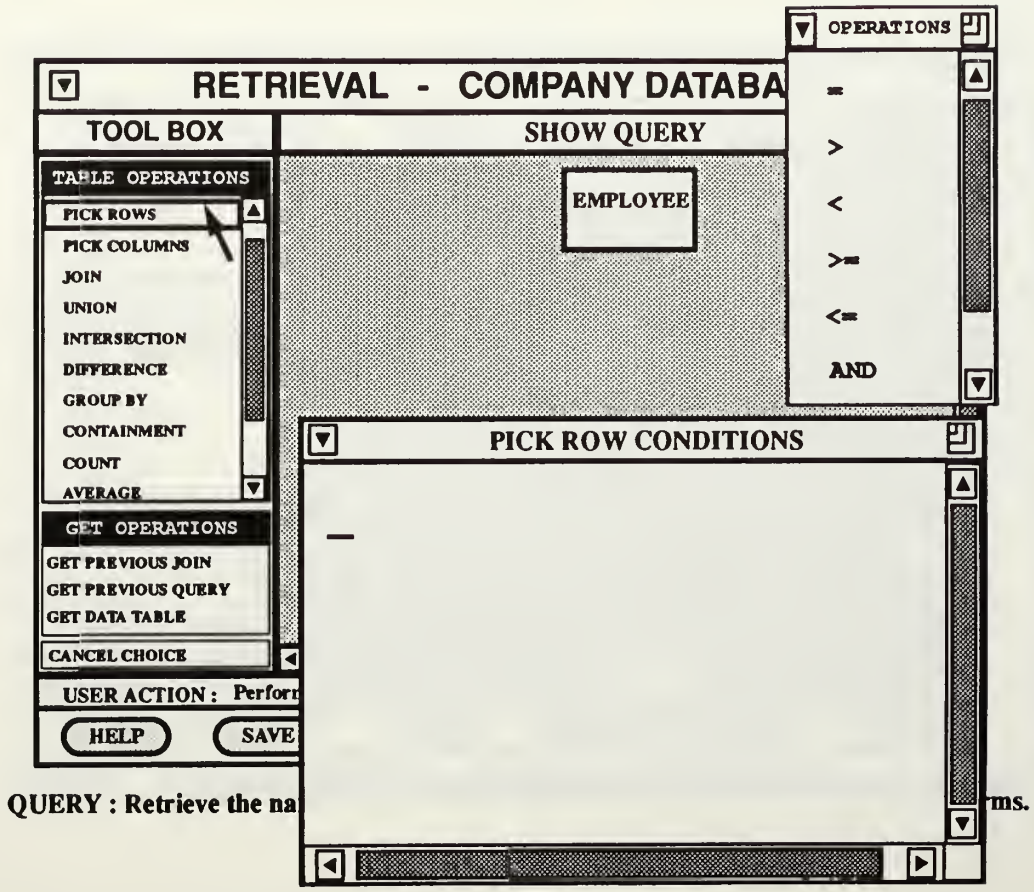


Figure 2.3 Simple Query With Direct Use of Multimedia Data

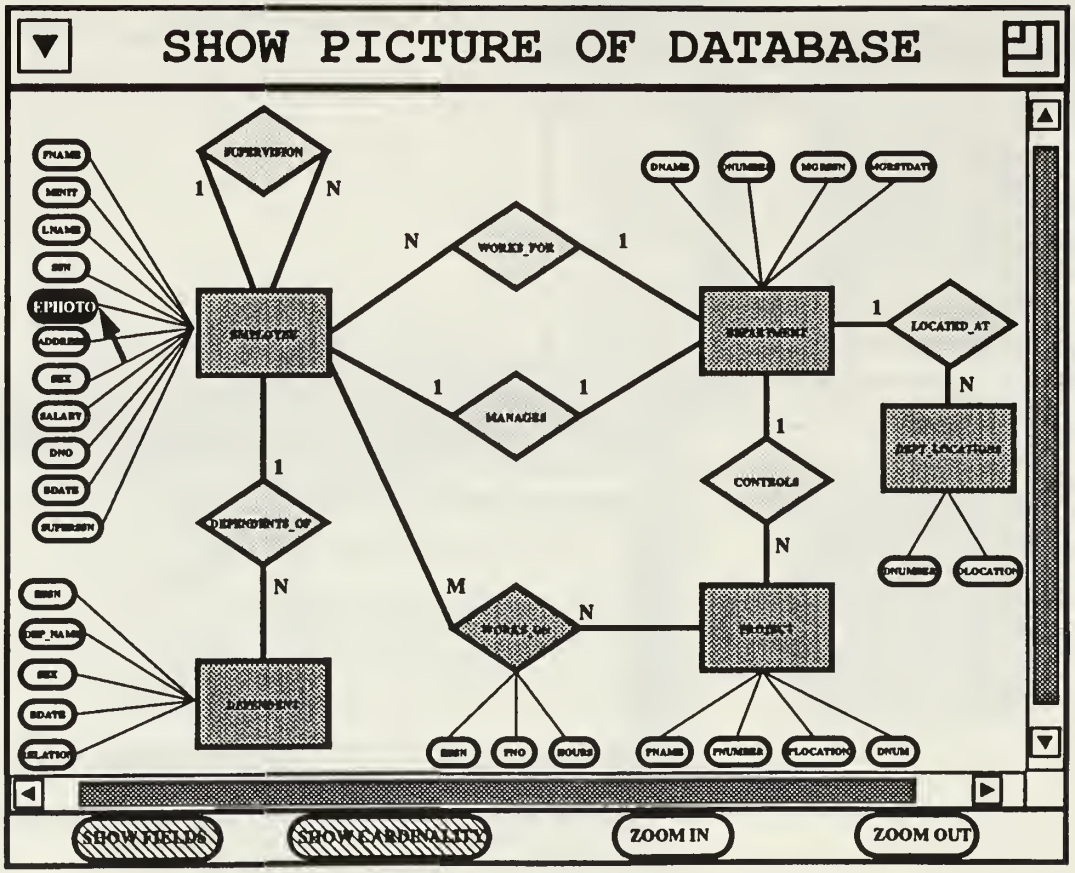


Figure 2.4 Simple Query With Direct Use of Multimedia Data

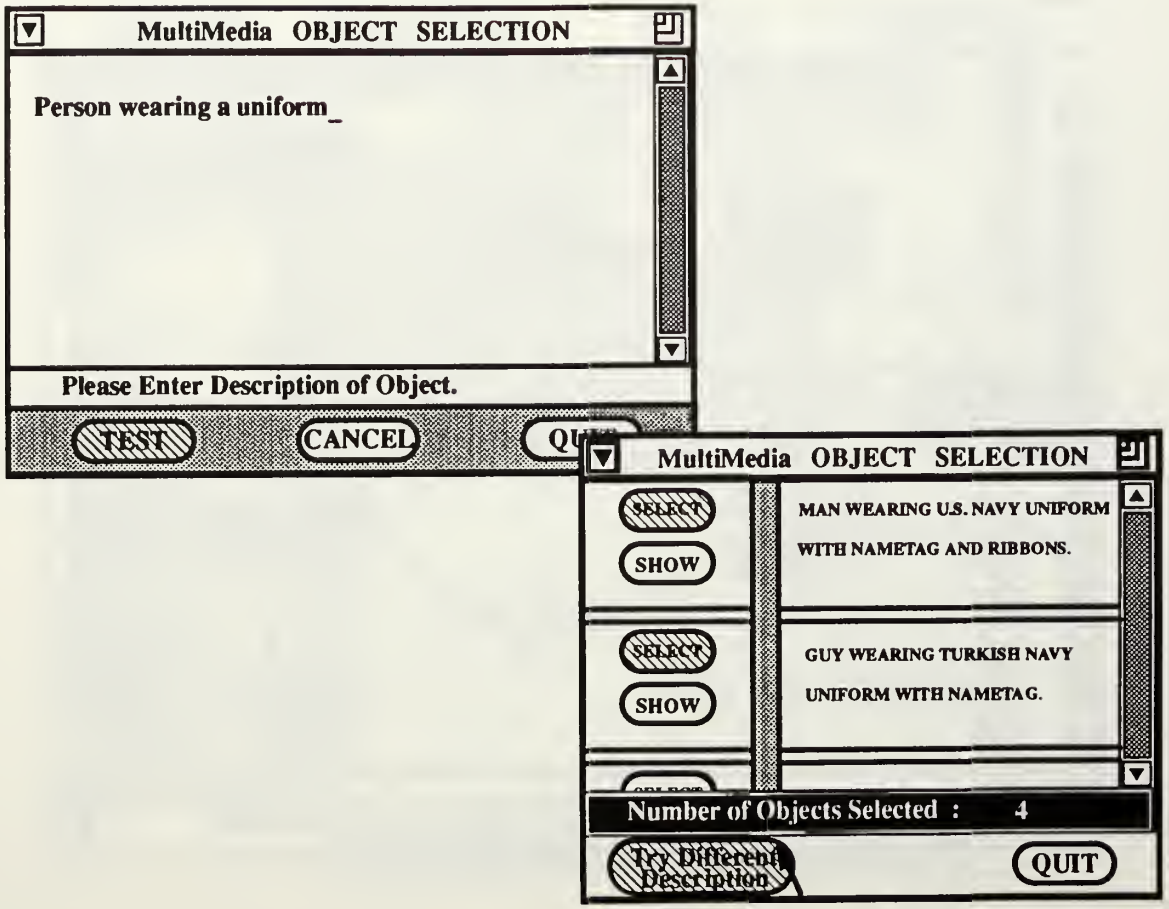


Figure 2.5 **Simple Query With Direct Use of Multimedia Data**

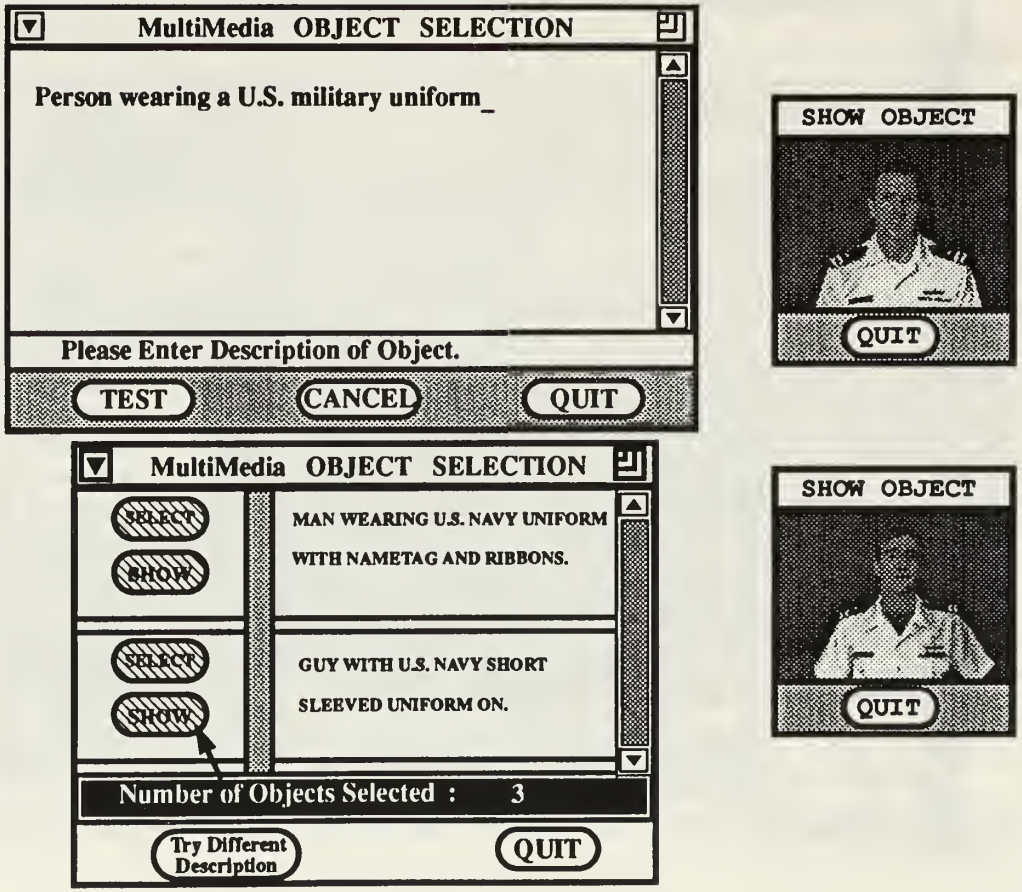


Figure 2.6 Simple Query With Direct Use of Multimedia Data

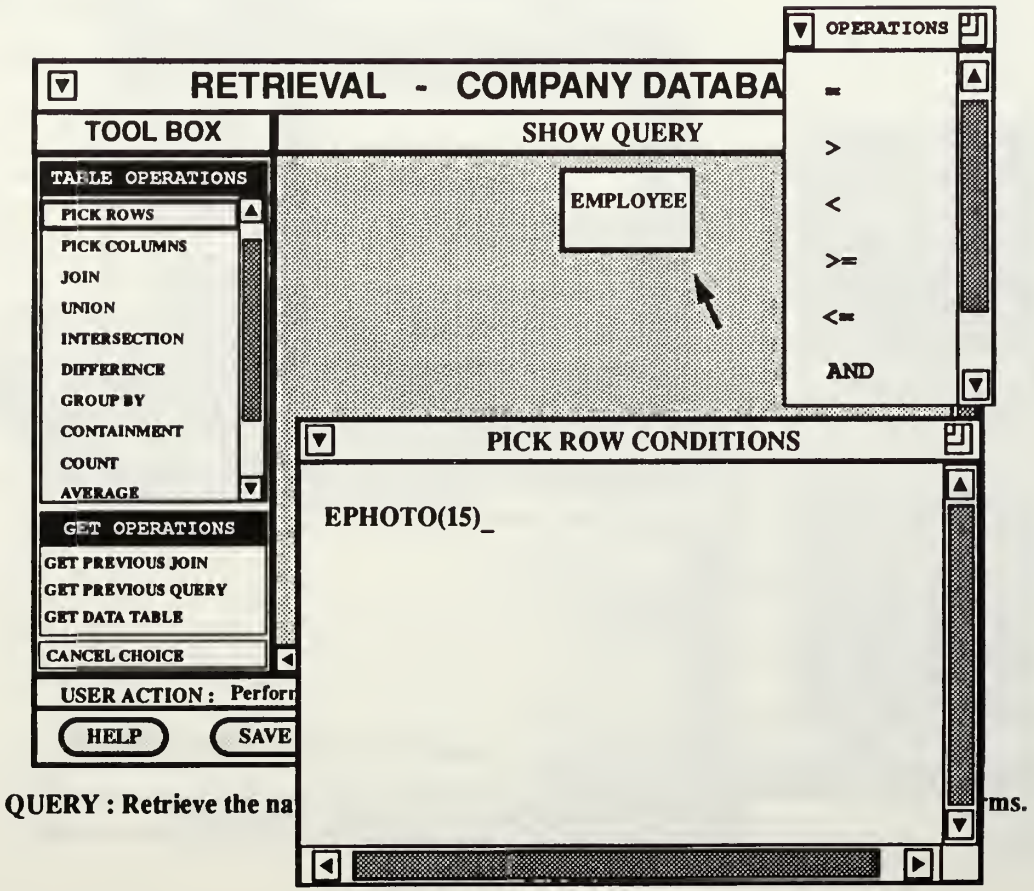
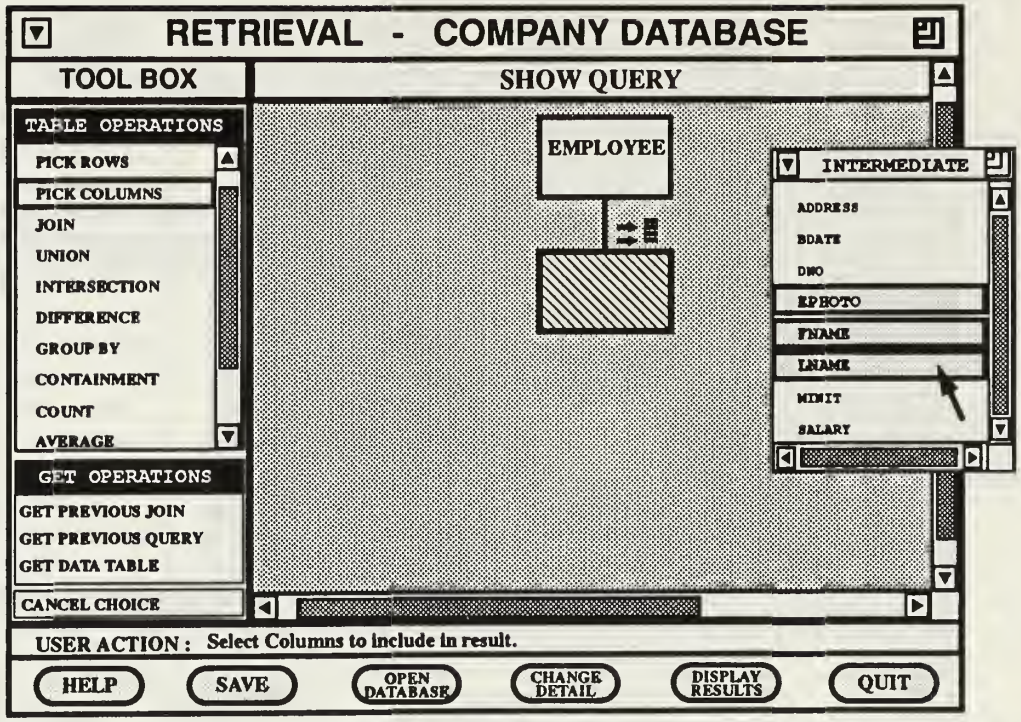
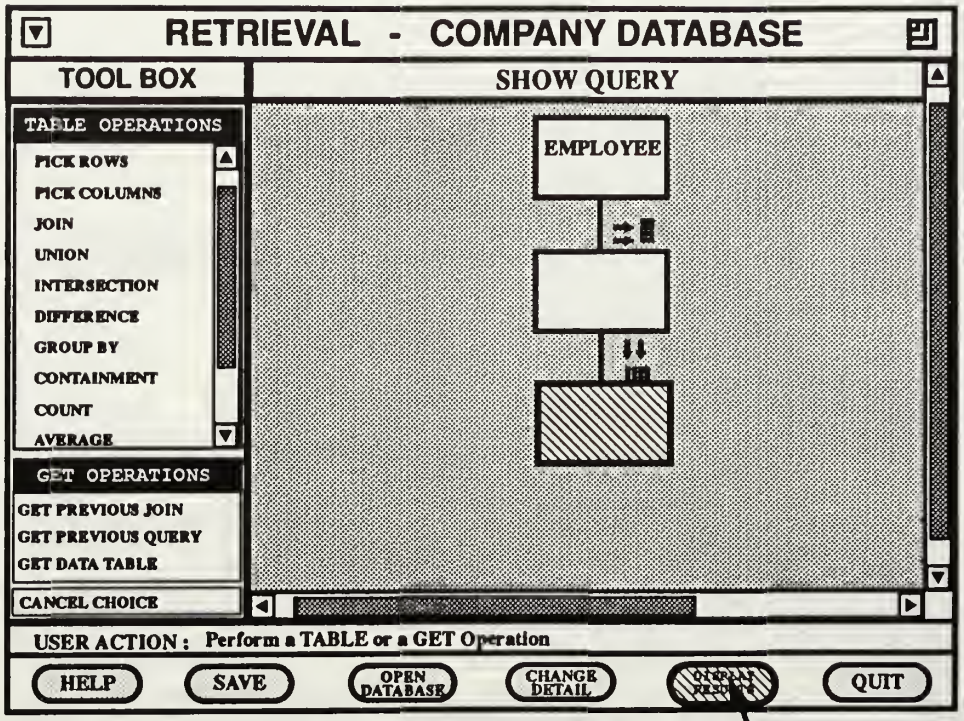


Figure 2.7 Simple Query With Direct Use of Multimedia Data



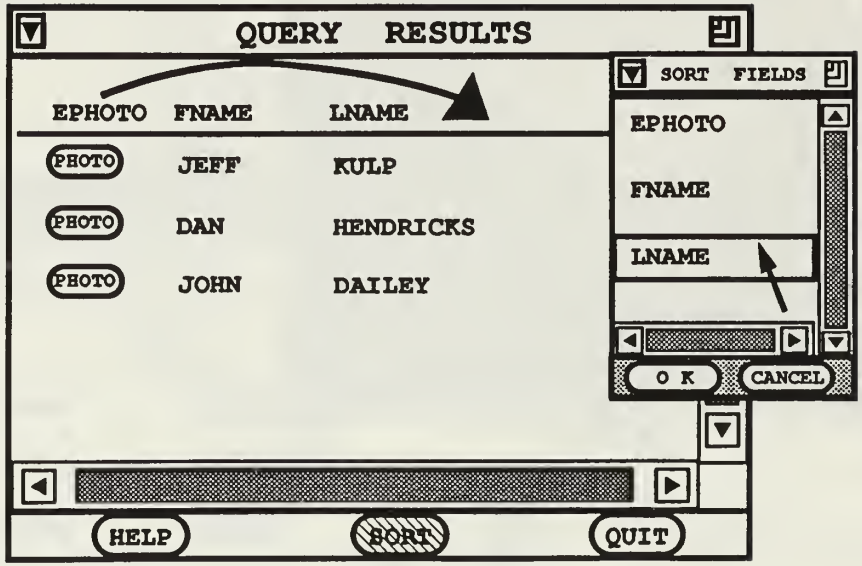
QUERY : Retrieve the name and photograph of employees wearing U.S. military uniforms.

Figure 2.8 Simple Query With Direct Use of Multimedia Data



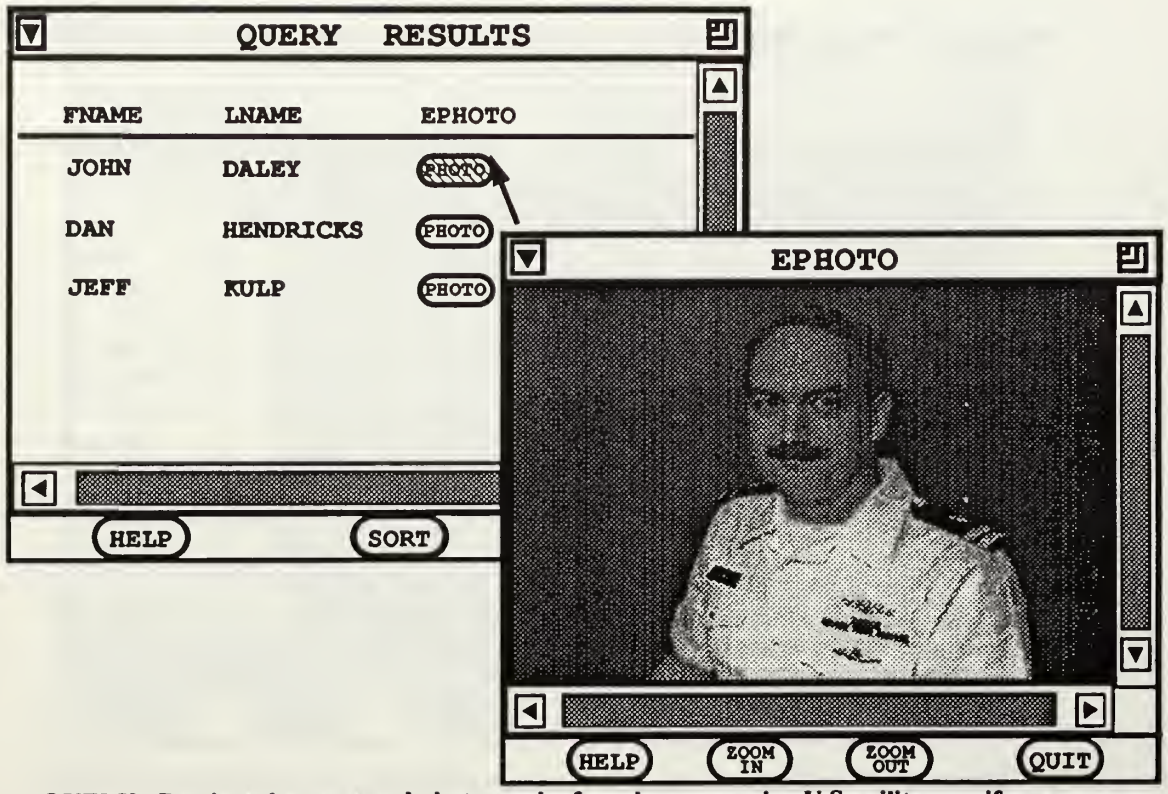
QUERY : Retrieve the name and photograph of employees wearing U.S. military uniforms.

Figure 2.9 Simple Query With Direct Use of Multimedia Data



QUERY : Retrieve the name and photograph of employees wearing U.S. military uniforms.

Figure 2.10 Simple Query With Direct Use of Multimedia Data



QUERY : Retrieve the name and photograph of employees wearing U.S. military uniforms.

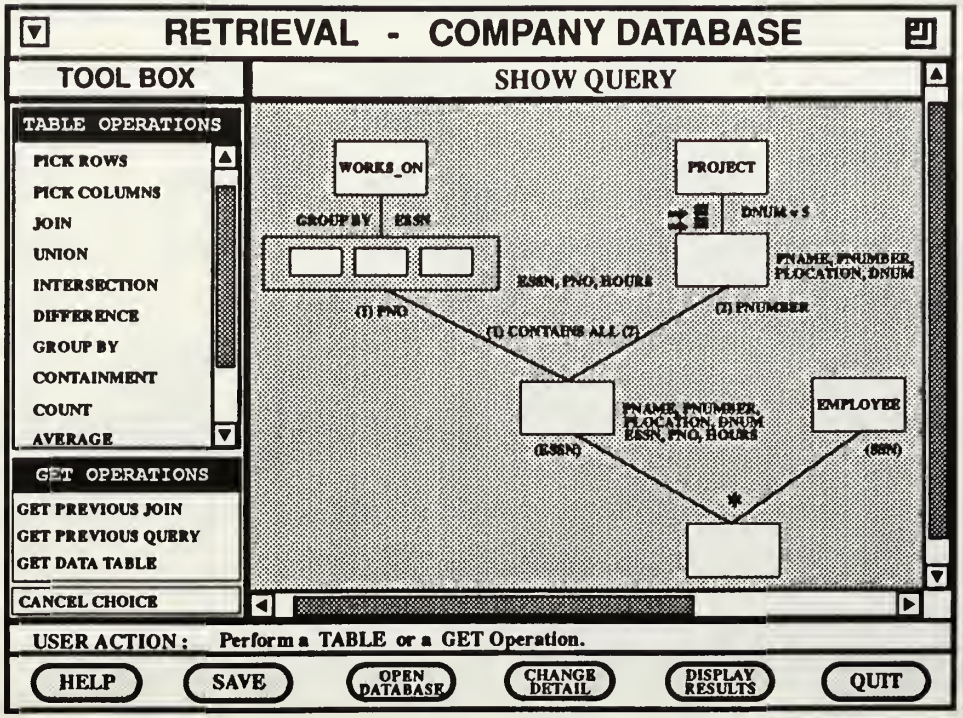
Figure 2.11 Simple Query With Direct Use of Multimedia Data

```
SQL   : SELECT   FNAME, LNAME
        FROM     EMPLOYEE
        WHERE    ( ( SELECT    PNO
                     FROM      WORKS_ON
                     WHERE     SSN = ESSN )
                     CONTAINS
                     ( SELECT  PNUMBER
                       FROM    PROJECT
                       WHERE   DNUM = 5) )

SQL   : SELECT    FNAME, LNAME
        FROM      EMPLOYEE
        WHERE     NOT EXISTS
                  ( SELECT    *
                    FROM      WORKS_ON  B
                    WHERE     ( B.PNO IN ( SELECT     PNUMBER
                                           FROM       PROJECT
                                           WHERE      DNUM = 5) )
                              AND
                              NOT EXISTS    ( SELECT   *
                                              FROM     WORKS_ON  C
                                              WHERE    C.SSN = SSN
                                                       AND C.PNO = B.PNO ) )
```

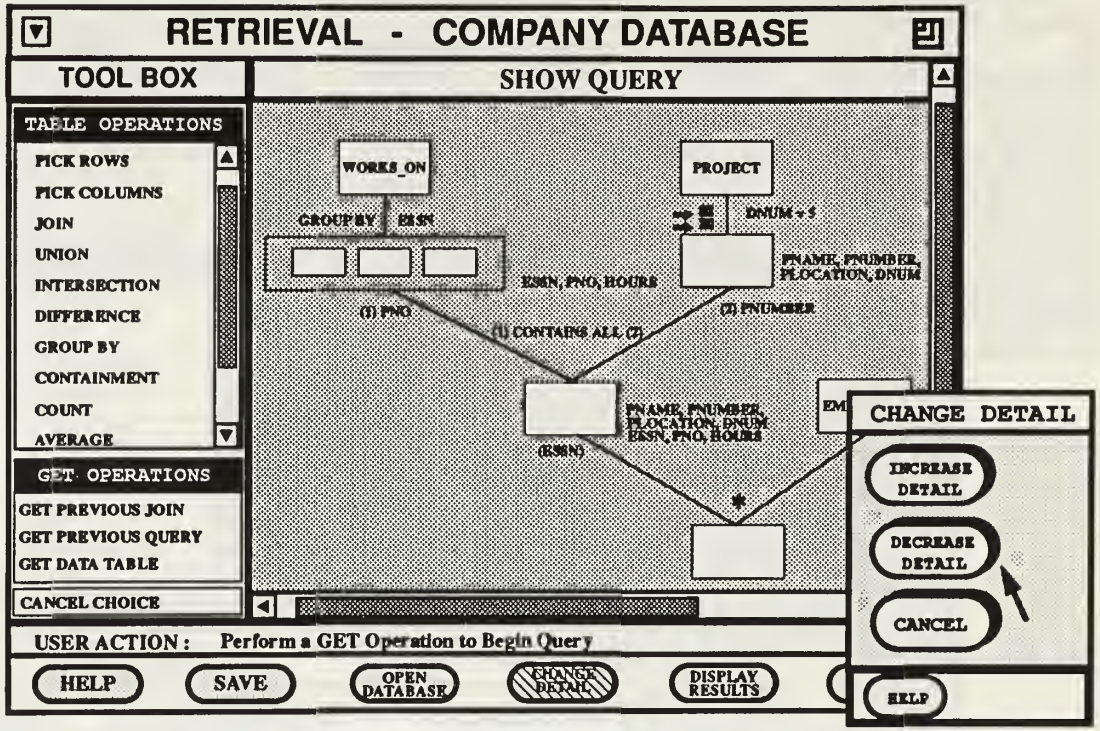
QUERY : Find the names of employees who work on all the projects controlled by department 5

Figure 3.1 Complex Query



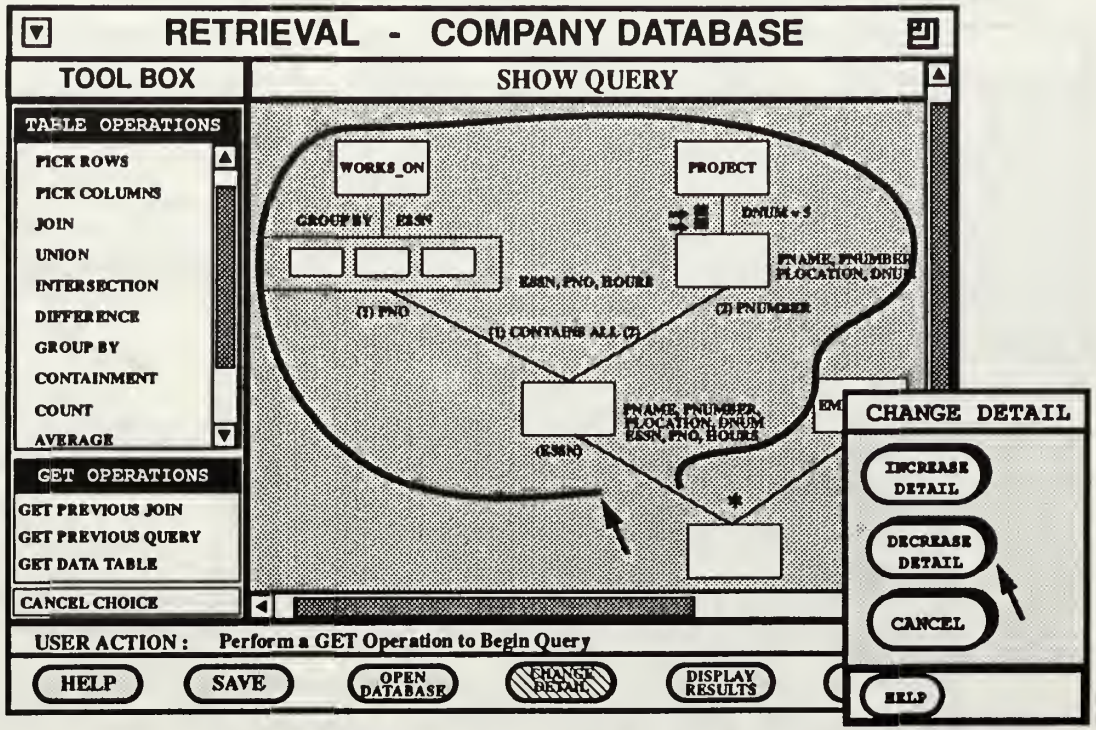
QUERY : Find the names of employees who work on all the projects controlled by department 5

Figure 3.2 Complex Query



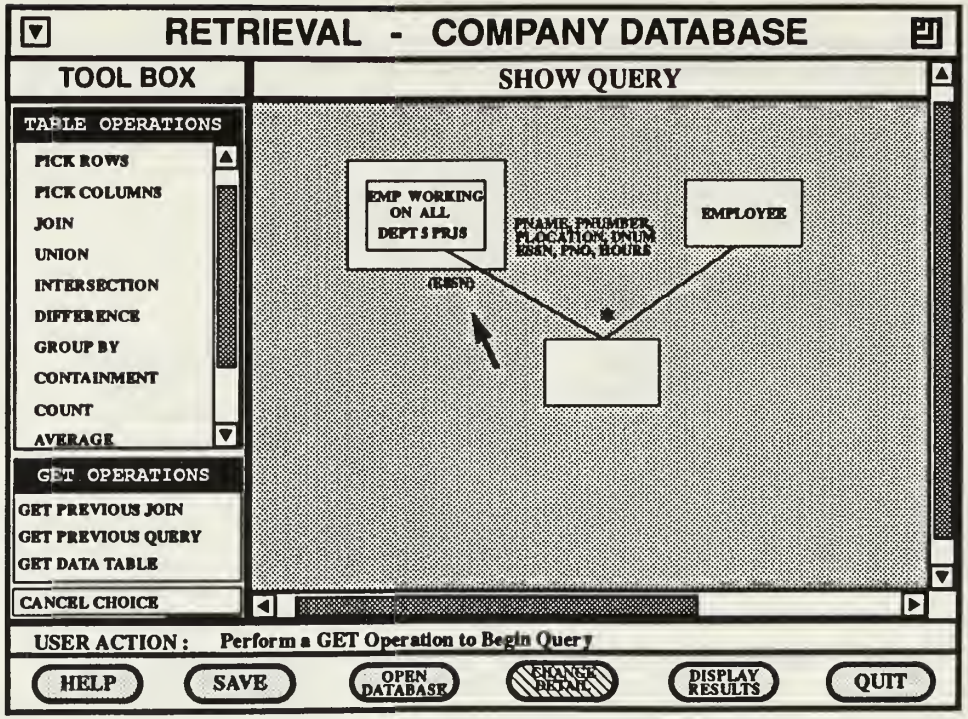
QUERY : Find the names of employees who work on all the projects controlled by department 5

Figure 3.3 Complex Query



QUERY : Find the names of employees who work on all the projects controlled by department 5

Figure 3.4 Complex Query



QUERY : Find the names of employees who work on all the projects controlled by department 5

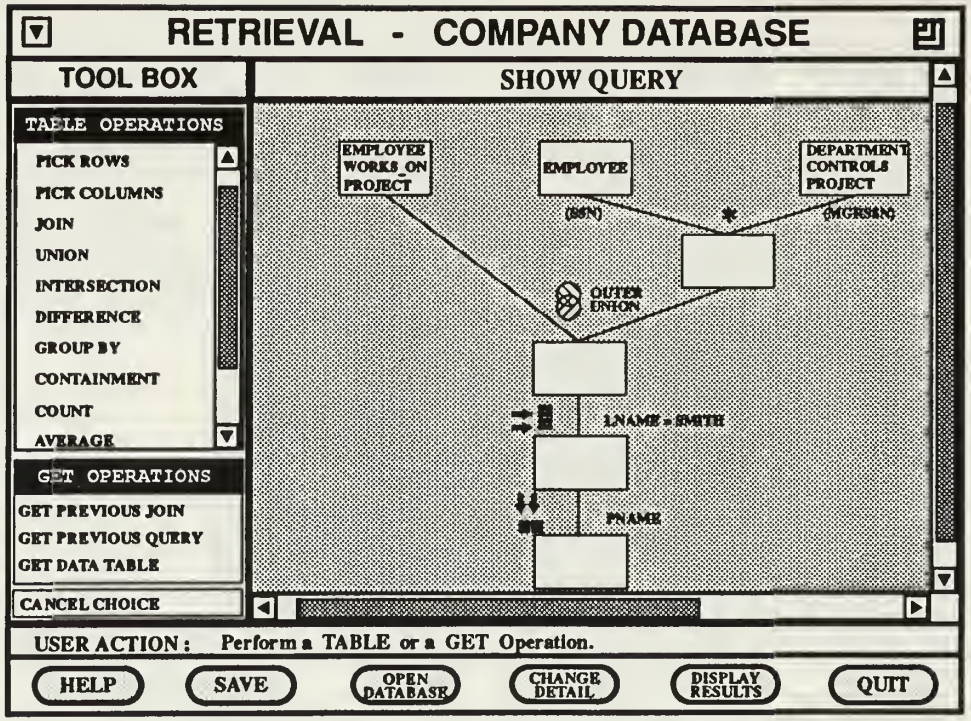
Figure 3.5 Complex Query

```
SQL   : ( SELECT   PNAME
        FROM     PROJECT, DEPARTMENT, EMPLOYEE
        WHERE    DNUM = DNUMBER AND MGRSSN = SSN AND LNAME = 'SMITH' )
          UNION
        ( SELECT   PNAME
        FROM     PROJECT, WORKS_ON, EMPLOYEE
        WHERE    PNUMBER = PNO AND ESSN = SSN AND LNAME = 'SMITH' )
```

```
SQL   : SELECT   DISTINCT PNAME
        FROM     PROJECT
        WHERE    PNUMBER IN ( SELECT   PNUMBER
                              FROM     PROJECT, DEPARTMENT, EMPLOYEE
                              WHERE    DNUM = DNUMBER AND MGRSSN = SSN
                                       AND LNAME = 'SMITH'
                             OR
                             PNUMBER IN ( SELECT   PNO
                                          FROM     WORKS_ON, EMPLOYEE
                                          WHERE    ESSN = SSN AND LNAME = 'SMITH'
```

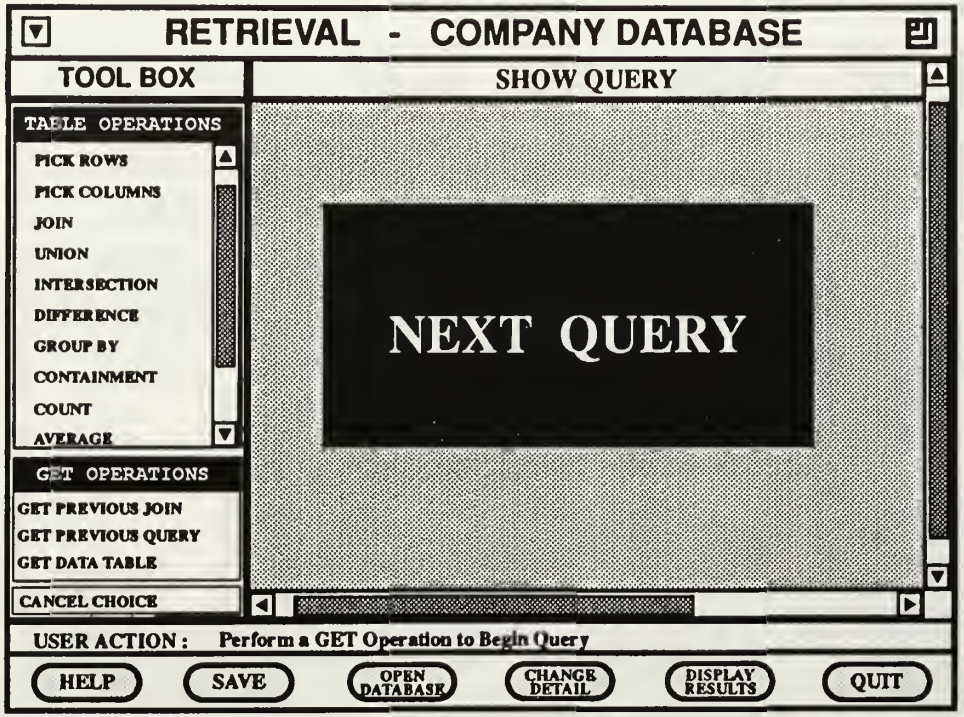
QUERY : List the project names for projects that involve an employee whose last name is 'SMITH' as a worker or as a manager of the department that controls the project.

Figure 4.1 Complex Query



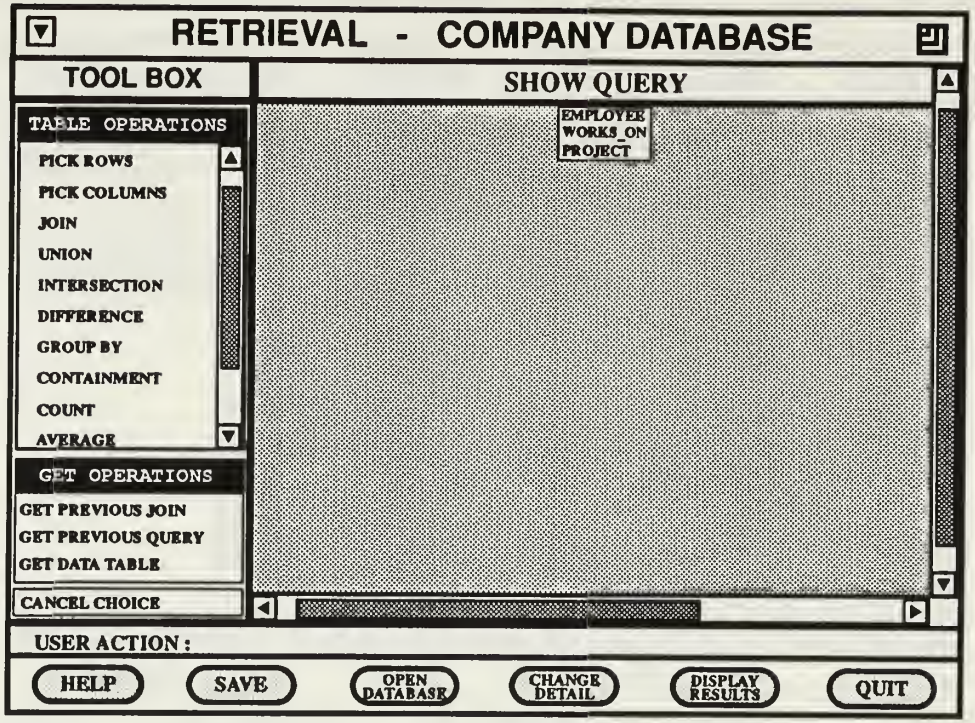
QUERY : List the project names for projects that involve an employee whose last name is 'SMITH' as a worker or as a manager of the department that controls the project.

Figure 4.2 Complex Query



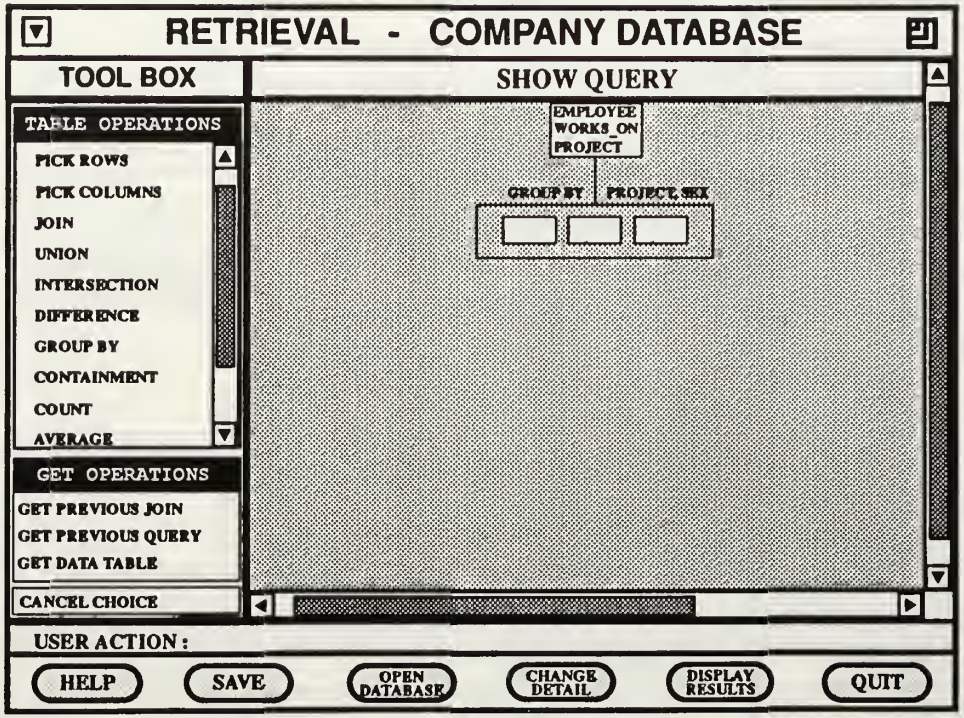
QUERY : For each project, find the average salary, the average salary of males and females, and the number of males and females.

Figure 5.1 Aggregate Functions



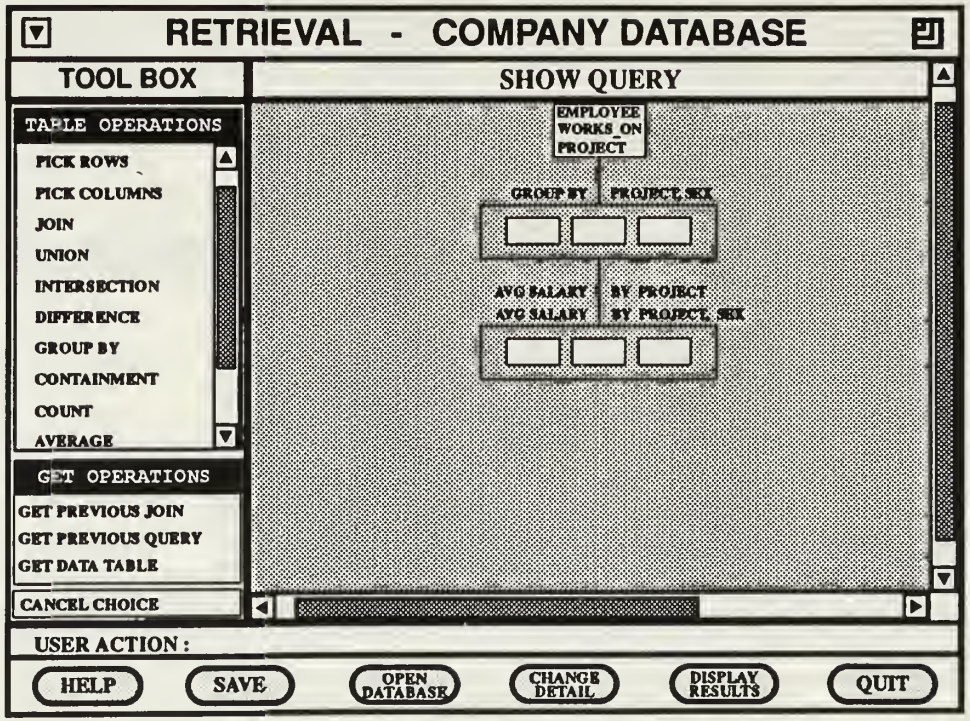
QUERY : For each project, find the average salary, the average salary of males and females, and the number of males and females.

Figure 5.2 Aggregate Functions



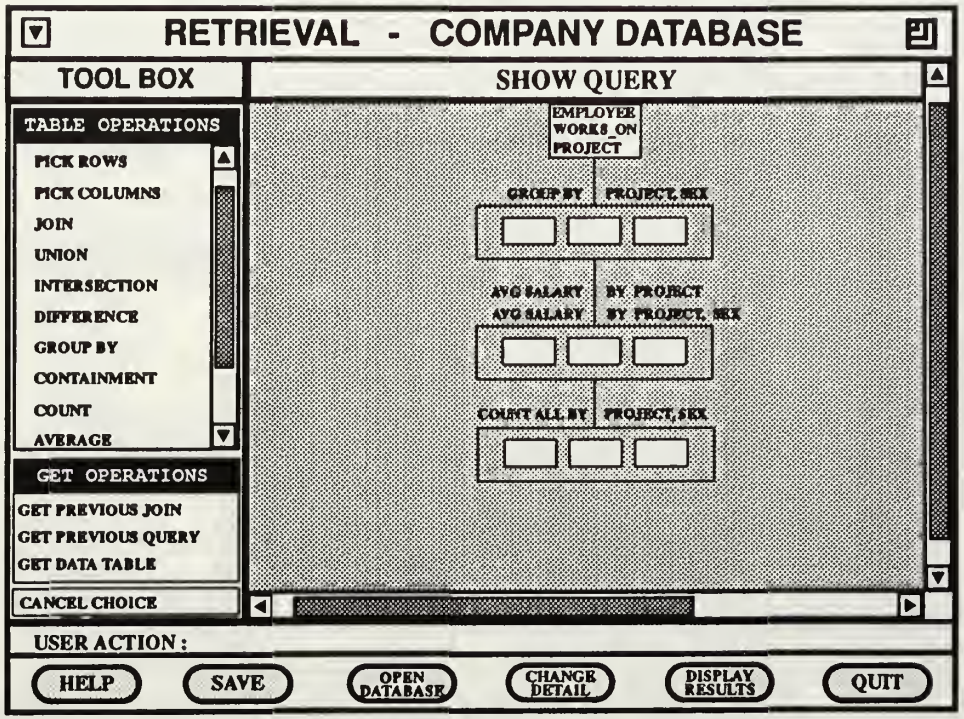
QUERY : For each project, find the average salary, the average salary of males and females, and the number of males and females.

Figure 5.3 Aggregate Functions



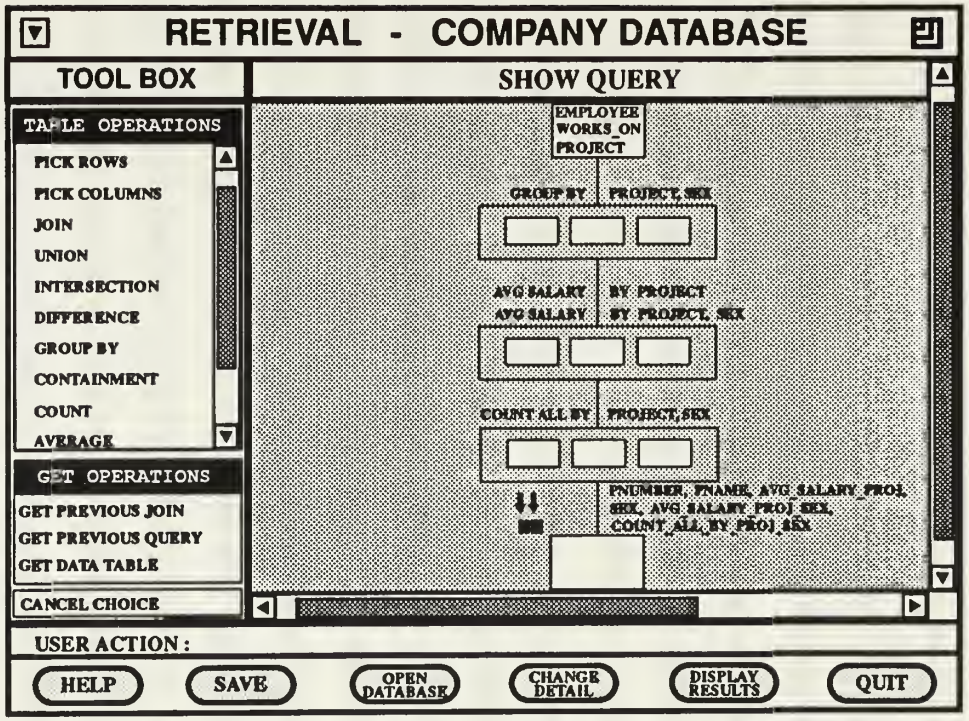
QUERY : For each project, find the average salary, the average salary of males and females, and the number of males and females.

Figure 5.4 Aggregate Functions



QUERY : For each project, find the average salary, the average salary of males and females, and the number of males and females.

Figure 5.5 Aggregate Functions



QUERY : For each project, find the average salary, the average salary of males and females, and the number of males and females.

Figure 5.6 Aggregate Functions

V. CONCLUSIONS

A. APPLICABILITY OF APPROACH

The graphical user interface developed in this thesis has been designed as part of a research project on multimedia database. This fact does not imply multimedia specific limitations on the applicability of the approach. The proposed database GUI is applicable to any relational DBMS. The goals in researching and designing this interface included an aim at ensuring general relational database applicability.

The storage, retrieval and manipulation of multimedia objects can be handled nicely within a relational framework (LUM89). The intent of the multimedia database project is to enable multimedia to be handled in a manner identical to structured data. Specific multimedia data are no more than values belonging to an abstract data type (e.g., photo). The values simply fall within the domain of an attribute of a relational tuple. The general user interface for a relational DBMS need only add the special capabilities necessary for working with multimedia data in order to be a functional multimedia database user interface. In other words, the functionality of a user interface for a relational multimedia database is a superset of the functionality required for a traditional relational DBMS. The user interface proposed in this thesis provides such a superset.

B. STRENGTHS AND LIMITATIONS OF APPROACH

1. Strengths of Approach

The Query Management Facility (QMF) presented in Chapter 4 provides the reader with a view of a good user interface for enabling a user to express queries against a relational database. The number one strength in the approach presented is its foundation in sound user interface principles and its use of the latest tools available and practical from currently available technology. The features which make up the interface have been carefully considered to provide the best possible interface. The user interface meets all of the criteria presented in Section A of Chapter 3, by which we can evaluate a user interface. Compliance with each of these criterion is discussed in the corresponding section below.

a. Criterion 1 : The Proposed DBMS Graphical User Interface (GUI) Must Constitute an Improvement Over Existing DBMS Interfaces

The proposed user interface constitutes an improvement over existing database user interfaces. This is the most difficult criterion to argue due to its inherent vagueness. While we do not have a standard metric by which we can measure the proposed interface against other interfaces we can make some general observations. These observations provide a reasonable measure of confidence that the interface succeeds in meeting this criterion.

User interfaces seen on databases today were never designed from the ground up to take advantage of today's latest hardware, software, and user interface capabilities. Most databases in use today have been around for some time. When originally implemented these databases used the hardware and software technology available at the time. This included linear query languages as well as a text based interaction style. Then hardware and software as well as experience with man-machine interaction began to evolve. This included such things as high resolution bit-mapped screens, the wide-spread use of the mouse, faster processors, windowing technology, graphics software, and graphical user interface capabilities and concepts. The companies with existing database products had large investments tied up in their existing software as well as large already established customer bases. Neither of these factors are conducive to rapid or radical evolution of software. This scenario lead to an incremental inclusion of pieces of technology to at leawt give the appearance with version upgrades, of user friendly, up-to-date user interfaces. As an example, there seemed to be a time when many of the database user interfaces were changing in order to introduce an interface with mouse controlled selection from pull-down or pop-up menus. The point is that user interfaces available for databases today were not designed from the ground up to take advantage of the latest and ever improving technology and user interface capabilities available. The user

interface proposed in this thesis is a definite exception. The observation that this interface is purposefully designed to integrate and take advantage of the latest and greatest capabilities is reason to argue that it is an improvement over existing interfaces.

b. Criterion 2 : The Proposed DBMS GUI Must Include the Integration of Applicable GUI Concepts and Capabilities

The user interface presented in this thesis includes the integration of those GUI concepts and user interface capabilities which contribute in a positive way to the interface. By surveying the literature available on this subject as well as the hardware and software available the goal was to consider all technology available. The absence in the user interface of such things as voice recognition and voice synthesis is not by careless omission. The inclusion of windowing techniques as well as the use of the mouse result from considering their usefulness for the task. These as well as many other user interface capabilities were considered. Only those capabilities contributing in a positive way were chosen to be integrated into the user interface.

c. Criterion 3 : The proposed DBMS GUI Must Support A Real-World to Database Mapping Mechanism

The proposed user interface supports a real-world to database mapping. The pictorial view of the database provided to the user enables him to gain a comfortable feel for the structural content of his database. The increased

detail which is available at the users request is helpful in responding to the users need for more detailed information. The actual values in the database are easy to gain access to and database browsing is encouraged. Chapter 3 provides a more detailed explanation of the features which contribute to meeting this criterion.

d. Criterion 4 : The proposed DBMS GUI Must Support Flexible Expression of Query

The user interface presented in this thesis supports the flexible expression of the users' query. This is a key advantage of the "low-level, simple operation" approach taken in this interface. Early database user interfaces were non-graphical in nature. In such an environment anything required of the user at a low-level necessarily implied undesirable amounts of tedious work (i.e., key-punching). It is no wonder that in the non-graphical environment designers of interfaces and query as well as general purpose programming languages decided to move away from anything low-level.

Things have changed. Graphical user interfaces are here to stay. There must be a re-thinking of earlier decisions to move away from low-level. In some cases earlier decisions are still valid. GUI's do not mean we should move from high-level procedural languages such as Ada, back to assembly language. In the case of database queries GUI's do require a re-thinking. With database queries there are advantages to the low-level approach. The approach, for example, facilitates the

users mental process of expressing the complex query. Further explanations of advantages are included in Chapter 3. The disadvantage of low-level query expression is no longer valid. It used to be that query expression at a level similar to the relational algebra implied tedious, time consuming work. Now with GUI capabilities, this process is greatly accelerated and made more easy. The user now can in many cases point and shoot, using the mouse. He can point at graphical objects on the screen and at menu options. There is seldom a need to key in anything. Additionally, the entire process is accompanied with meaningful pictures which give the user immediate and meaningful feedback. These factors combine in making low-level query expression a useful and highly beneficial technique. At this low-level, a great deal of flexibility is provided to the user to express the query in whatever way he thinks about the query.

e. Criterion 5 : The proposed DBMS GUI Must Comply With Known User Interface Principles

The proposed user interface complies with known user interface principles. Section A of Chapter 2 presents a couple lists which provide a summary of the types of principles which are commonly considered and applied to user interface design. These lists include such things as consistency, informative feedback, simple error handling, easy reversal of actions, reduction in short-term memory load, intuitiveness, flexibility and use of visual cues. Compliance

with these and other such principles is attempted throughout the user interface which is presented. In some cases the attempts are specifically mentioned and in many other cases, it is obvious.

f. Criterion 6 : The Proposed DBMS GUI Must be Extensible

The user interface presented in this thesis is extensible to the degree a specification can be extensible. The specification intentionally makes no reference to a particular windowing standard, window manager, or user interface standard. Extensibility is more important to consider when making implementation decisions. This specification was carefully constructed so as to make it as implementation independent as possible. The specification lends itself towards the modularity required in order to ensure an extensible implementation.

2. Limitations of Approach

The user interface presented in this thesis is constrained by its applicability to the relational data model. There are other data models such as the hierarchical, the network and the object oriented data models. The decision to assume the relational data model is not to say that these other models are not useful and in some cases even more suitable for certain applications. The research findings presented in Chapter 3 are for the most part applicable to a query management facility regardless of the underlying data

model. Regardless of the data model upon which a database is built, the user must understand the relationship between the real world and his specific database. The user must also be given the best means of expressing his complex queries. The user must also be given the tools to allow graphical manipulation and simple operations to ensure his efficient and correct formulation of queries. These, as well as many of the other objectives in designing a relational query facility, carry over to query facilities for other data models. The point is that the specific interface proposed is constrained in its applicability but the principles underlying its design are not.

Another limitation is not so much a limitation of this approach as it is specified, but a limitation created by the lack of an overall user interface specification. Prior to embarking on the query specification aspect of this thesis, it was felt that the query specification portion of a database user interface involved the biggest and most difficult challenges. It is for this reason that this portion of the interface was tackled first. The limitation stems from the fact that the entire database user interface is not yet completely specified. This leaves the possibility that specification and implementation of other related user interface components of the database may run into constraints, the solutions of which may in some way impact the design or implementation of the Query Management Facility. This idea

raises the notion of whether the design specification of a component of a larger system can be considered complete until all the components are specified. It would be a comfortable feeling to know that in the end all the components will integrate nicely together. Perhaps this being part of a research project mitigates this concern.

C. FUTURE WORK

1. Implementation of the Proposed Query Management Facility

This thesis presents a specification for the user interface for a query management facility of a relational database. It remains to implement this specification. The specification is specific in idea but not in the detail of implementation. This lack of specifics is to enable the implementor as much flexibility as possible in choosing the software and hardware tools to use as well as in putting his own touch on the finished product.

Amongst the more important implementation decisions are those relating to software. These include the following: What operating system and what version operating system will be used(e.g., newest version of Unix)?, What programming language will be used (e.g.,K&R C, ANSI C, C++)?, What windowing system will be used (e.g.,X-window) (MANDELKERN90, SHEIN90)?, What high-level development environment if any, will be used (MANDELKERN90, SHEIN90)?, What look and feel user

interface standard will be used (e.g.,Open Look, Motif) (BOWEN90, HELLER90a, HELLER90b)?

The implementor must also consider various hardware decisions. Included among these are: Is this to be a multi-platform system ?, What Input/Output devices will work with what platforms?, What software will work on what platforms ?

A number of trade-offs will have to be made. These trade-offs should be made in a conscious manner and documented for future designers and maintainers. Considerations such as portability, availability, compatibility, affordability, learnability, and supportability will likely force some painful and less then ideal decisions.

2. Design and Implementation of Remaining User Interface Components for Database

As mentioned already there are three major facilities which were considered (i.e.,SMF,QMF,RMF). Of these three, only the QMF has been considered in detail. This leaves the task of creating a detailed specification of the other two facilities.

A good user interface must contain a complete set of good components. Each component must lend itself synergistically to the aggregation of a complete package. There should not be a query management facility which works via graphical direct manipulation and a report management facility which is based on hierarchical textual based menus. Regardless of how good each component is, they just do not fit together. The following list contains the common elements

which must be consistent throughout the database user interface :

1. Maximum use of the graphical direct manipulation paradigm.
2. Design which lends itself to a windowing environment.
3. Design consistent with sound user interface principles (e.g., those mentioned in Section A of Chapter 2).
4. Design which includes the ability to work with the Relational Data Model.

An interface component which abides by these guidelines will integrate nicely with the query management interface which has been proposed.

3. Continuing Incorporation of Evolving Technology

The future certainly will see continued advances in hardware and software technology as well as new and improved techniques for man-machine interface. The extensibility mentioned in Criterion 6 of Chapter 3 is intended to enable the user interface to grow and change along with this advancing technology.

LIST OF REFERENCES

Agrawal R., Gehani N.H., Srinivasan J., *OdeView : The Graphical Interface to Ode*, pp. 34-43, Proc. ACM SIGMOD 1990, International Conference on Management of Data, Atlantic City, May, 1990.

Anderson T.L., Ecklund E.F., Maier D., *The PROTEUS Bibliography : Representation and Interactive Display in Databases*, SIGMOD Record, Vol 15, No 3, September, 1986.

Bowen B., *Open Look vs. Motif, The Battle for GUI Dominance*, Sun Tech Journal, Vol 3, Num 6, December 1990.

Brown J.R., Cunningham S., *Programming the User Interface, Principles and Examples*, Published by John Wiley and Sons, 1989.

Bryce D., Hull R., *SNAP : A Graphics Based Schema Manager*, pp. 151-164, Proc. IEEE 2nd International Conference on Data Engineering, Los Angeles, California, February, 1986.

Codd E.F., *Relational Completeness of Database Sublanguages, In Database Systems*, pp. 65-98, Prentice-Hall, 1972.

Comaford C., *Graphical User Interfaces - Keep Them Sleek and Simple*, Info World, March, 1991.

Elmasri R.A., Larson J.A., *A Graphical Query Facility for ER Databases*, Proc. 4th International Conference, Entity-Relationship Approach, Chicago, October, 1985.

End User Interfaces, Introduction, SIGMOD Record, Vol 18., No 1, p. 23, March, 1989.

Goldman K.J., Goldman S.A., Kanellakis P.C., Zdonik S.B., *ISIS: Interface for a Semantic Information System*, pp. 328-342, ACM SIGMOD Proceedings of the International Conference on the Management of Data, Austin, Texas, May, 1985.

Gyssens M., Paredaens J., Gucht D.V., *A Graph-Oriented Object Model for Database End-User Interfaces*, pp. 24-33, Proc. ACM SIGMOD 1990, International conference on Management of Data, Atlantic City, May, 1990.

Heller D., *Look & Feel*, Sun Tech Journal, Vol 3, Num 6, December 1990(a).

Heller D., *Objects and Widgets*, Sun Tech Journal, Vol 3, Num 6, December 1990(b).

Keim D., Kim K.C., Lum V.Y., *A Friendly and Intelligent Approach to Data Retrieval in a Multimedia Database*, Tech Report NPSCS-91-010, Computer Science Department, Naval Postgraduate School, Monterey, California, March, 1991.

Kim K.C., Lum V.Y., *Towards Intelligent Data Retrieval in Multimedia Databases*, Tech Report NPSCS-91-009, Computer Science Department, Naval Postgraduate School, February, 1991.

Kuntz M., Melchert R., *Pasta-3's Graphical Query Language: Direct Manipulation, Coopertive Queries, Full Expressive Power*, pp. 97-105, Proc. of 15th International Conference on Very Large Data Bases, Amsterdam, The Netherlands, August, 1989.

Leong M.K., Sam S., Narasimhalu D., *Towards a Visual Language for an Object Oriented Multi-Media Database System*, Visual Database Systems, Elsevier Science Publishers B. V. (North-Holland), 1989.

Lum V.Y., Meyer-Wegner K., *A Multimedia Database Management System Supporting Contents Search In Media Data*, Technical Report, NPS52-89-020, Computer Science Department, Naval Postgraduate School, Monterey, California, March, 1991.

Maier D., Nordquist P., Grossman M., *Displaying Database Objects*, pp. 15-30, Proc. of the 1st International Conference on Expert Database Systems, Charleston, South Carolina, April, 1986.

Mandelkern D., *A Guide to High-Level User Interface Development Tools*, Sun Expert, Vol 1, Num 3, January, 1990.

Meyer-Wegner K., Lum V.Y., Wu C.T., *Image Management in a Multimedia Database System*, Proc. of the IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems, Tokyo, Japan, 3-7 April, 1989.

Miyao J., Hirakawa N., Kikuno T., Yoshida N., *Design of a Form Interface Language in a Database System Aide*, pp. 26-27, Proc. of the IEEE Workshop on Languages for Automation, Vienna, Austria, August, 1987.

Miyao J., Tominaga K., Kikuno T., Yoshida N., *Design of a High Level Query Language For End Users*, pp. 27-29, Proc. of the IEEE Workshop on Languages for Automation, Kent Ridge, Singapore, August, 1986.

Rogers T.R., Cattell G.G., *Entity-Relationship Database User Interfaces*, Readings in Database Systems, Ed. by Stonebraker M., 1988.

Shein B., *Primal Screens*, Sun Expert, Vol 1, Num 3, January, 1990.

Shneiderman B., *Designing the User Interface*, Published by Addison-Wesley, 1987.

Shu N.C., *Visual Programming : Perspectives and Approaches*, IBM Systems Journal, Vol 28, No 4, 1989.

Smith S.L., Mosier J.N., *Guidelines for Designing User Interface Software*, Report prepared by Mitre Corporation for the USAF, ESD-TR-86-278, August, 1986.

Ullman J.D., *Principles of Database Systems*, Computer Science Press, 1982.

Wartik S.P., Penedo M.H., *FILLIN : A Reusable Tool for Form-Oriented Software*, IEEE Software 6, No 3, 1986.

Wegner L.M., *ESHER - Interactive, Visual Handling of Complex Objects in the Extended Second NF Database Model*, 1989, Visual Database Systems, Elsevier Science Publishers B. V. (North-Holland), 1989.

Wong H.K.T., Kuo I., *GUIDE : Graphical User Interface For Database Exploration*, pp. 22-32, Proc. of 8th Conference on Very Large Databases, Mexico City, 1982.

Wu C.T., Hsiao D.K., *Implementation of Visual Database Interface Using an Object Oriented Language*, Proc. of the IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems, Tokyo, Japan, 3-7 April, 1989.

Zdonik S.B., Maier D., *Interfaces*, Introduction to Chapter 8, Readings in Object-Oriented Database Systems, Morgan Kaufmann Publishers, Inc., 1990.

Zhang Z.Q., Mendelzon A.O., *A Graphical Query Language for Entity-Relationship Databases*, Entity-Relationship Approach to Software Engineering, Elsevier Science Publishers B. V. (North Holland), 1983.

Zloof M., *Query by Example*, pp. 431-438, AFIPS, Proceedings of the National Computer Conference, Volume 44, 1975.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center 2
Cameron Station
Alexandria, Virginia 22304-6145

Dudley Knox Library 2
Code 052
Naval Postgraduate School
Monterey, California 93943-5100

Center for Naval Analysis 1
4401 Ford Ave.
Alexandria, Virginia 22302-0268

John Maynard 1
Code 42
Command and Control Departments
Naval Ocean Systems Center
San Diego, California 92152

Dr. Sherman Gee 1
ONT-221
Chief of Naval Research
880 N. Quincy Street
Arlington, Virginia 22217-5000

Leah Wong 1
Code 443
Command and Control Departments
Naval Ocean Systems Center
San Diego, California 92152

Professor Vincent Y. Lum 2
Code CsLm
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

Capt. Charles B. Peabody 1
Unit 2, Bramber II,
Rt 16B, Rochester, NH 03867

Professor C. Thomas Wu 1
Code CsWu
Department of Computer Science
Naval Postgraduate School
Monterey, California 93943

Dr. Bernhard Holtkamp 1
University of Dortmund
Software Technology
P.O. Box 500
D-4600 Dortmund 50 / GERMANY

Professor Klaus Meyer-Wegener 1
University of Erlangen-Nuernberg
IMMD VI, Martensstr. 3,
5250 Erlangen / GERMANY

Commandant of the Marine Corps 1
Code TE 06
Headquarters, U.S. Marine Corps
Washington, D.C. 20380-0001

Thesis
P289 Peabody
c.1 Design of a graphical user interface for a multi-media DBMS : query management facility.

3 2768 00032016 2